Documentation of QuestDown

version 1.0

QuestDown is a language designed to write interactive events within maps generated with the hQuestBuilder software. This language is used to specify **Events** and **Actions** that occur during the game, executed with hQuestMaster. The following is a complete guide on how to write a Quest using QuestDown.

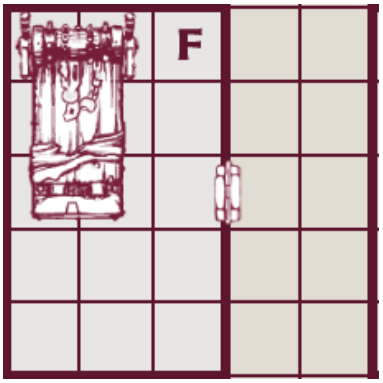
General principles of QuestDown

- QuestDown works via **Events** and **Actions**, specified via text strings.
- QuestDown is used by writing the Code in the Notes section in a map created via hQuestBuilder (https://www.hquestbuilder.com/)
- **Events** are commands that specify *when* something specific will happen. For example, when a door is opened, when a trap is activated, or when a specific monster is defeated.
- **Actions** are commands that specify *what* will happen. For example, through an action we can make a door open, move a monster, remove a piece of furniture from the board, etc. Basically, each Event is always associated with an action (by default) which is to display text on the screen.
- All automations that can be created in QuestDown *must* be specified via Events (when), which can then be associated with Actions (what).
- All Events are associated with a **Marker** (i.e. the letters or numbers that are in the maps of hQuestBuilder). It is always preferable to associate a Marker with a single Event, but it is possible for the same automations to be associated with several events.
- Some Events contain **Parameters**, i.e. some additional arguments that it is mandatory or optional to specify. For example, the ON_OPEN Event contains a mandatory parameter indicating the door or secret door to be opened.
- In events that add Monsters it is possible to specify **Attributes**, i.e. the characteristics of the monsters (movement, body points, mind points, etc.). Some of these Attributes are mandatory, while some (e.g. available Spells) are optional.
- Normally each Event only occurs once. In many events, however, it is possible to specify the **REPEAT** option after the Event name. Example [[(A), ON_STEP:REPEAT]]. If the REPEAT option is specified, this event repeats every time the trigger conditions occur.

An Example of QuestDown

Below is an example of a QuestDown command, with an associated hQuestBuilder map:

{[[(F), ON_ENTER_ROOM]] On the torture rack you found the corpse of the elf you were looking for}



By copying this code into the NOTES section of a hQuestBuilder map, the moment one of the heroes enters the room where marker F is, the specified text will appear on the screen (" On the torture rack you found the corpse of the elf you were looking for"). The event is not repeated, so the message will only appear the first time you enter the room.

This is a simple example of one of the many possible automations. The sections below will provide details on the general principles of QuestDown and a list of all available Events and Actions.
You can also consult a Glossary of terms.

QuestDown and official rules

QuestDown mostly follows the official HeroQuest rules. In some cases, however, more freedom has been deliberately left to allow House Rules or variants to be easily used (For example,. even the Barbarian or Dwarf may cast spells, or one may interrupt movement to search for a treasure and retrieve it, etc.). In these cases it is left to the players to stick to the correct rules. Monsters, on the other hand, will always behave according to the official rules.

General Event Syntax

An Event starts according to this general code pattern

{[[(marker), event1, event2, ...]] text and/or [[action1]] [[action2]],...}.

Example:

{[[(G),ON_OPEN]] As soon as you open the secret door a strange force pushes you towards the fireplace [[MOVE_HERO(K12)]] }.

In the following line, the diagram is reproduced by colouring the various parts in order to clarify the correspondence of each part of the code.

{[[(marker), event1; event2; ...]] text and/or [[action1]] [[action2]],...}.

Example

{[[(G),ON_OPEN]] As soon as you open the secret door a strange force pushes you towards the fireplace [[MOVE_HERO(K12)]] }.

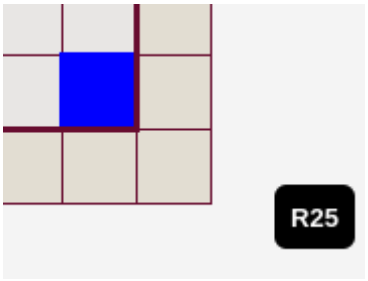
The ***marker*** represents the position on the map with which the event is associated, all markers allowed by hQuestBuilder, letters **A** to **Z**, a number from **1** to **20** or one of the arrows respectively called "FancyArrow", "Arrow", "inOut" can be used

After the marker, one or more Events separated by commas are listed.

- In many cases the exact location of the marker is not important. What is important is that it is in the room or section of the corridor where the event is to take place.
- However, there are some events (e.g. ON_STEP), which refer to the exact position of the marker, i.e. the associated actions only occur when something happens at the specific position of the marker.
- Finally, there are cases in which the position of the marker is completely irrelevant, because the Event has an effect on the game in general (e.g. ON_NEW_TURN), or because it must otherwise be specified where it will take effect (e.g. ON_OPEN).

Caution: If a marker is not present in the map, the events associated with that marker will never be activated.

When it is necessary to indicate a particular cell with an action or event, it must be indicated by a pair letters (A to S) and a number from 1 to 26; **it is possible to see what a cell's co-ordinate is directly from the hQuestBuilder by holding down Q and together D**, this will show a small box in the bottom right-hand corner of the browser containing the coordinates of the selected cell.



Any other text outside the {} will be ignored, so you can leave the original map notes, or add notes to the written QuestDown code.

List of Events

A list of all QuestDown Events is provided below. Remember, an Event is nothing more than a command indicating 'when' something will happen in hQuestMaster.

ON_START or START

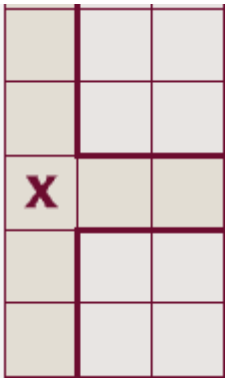
This Event is activated automatically at the start of the game and serves to indicate the starting position of the heroes. It is used as ON_START or ON_START(hero_number), or START or START(hero_number).

- Parameters: (**hero_number**) This is an optional value, if present, START will indicate the starting position of the hero number **hero_number** (e.g., if ON_START(2), then the 2nd hero; NOTE: the number of heroes corresponds to the one they were entered with on the 'Choose your team!' screen of the hQuestMaster website).
  If absent, it will indicate the position of the first hero, the second will be placed to his right, the third below and the fourth to the right of the third.

If there is no space to place all heroes in 4 separate cells, they will all be placed on the same cell.

- Example 1: **{[[(X), START]]}**
- Example 2: **{[[(X), ON_START]]}**
- Example 3: **{[[(X), ON_START(3)]]}**

In all these examples the heroes will start from the square where the X marker was placed.



The first two examples are equivalent.
In Example 4, the command ***{[[(X), ON_START(3)]]}*** *specifies that the first hero entered as 3rd in the list of heroes making up the team will start from the box with marker X. For example, in the case of the team below, this would be the Elf.*

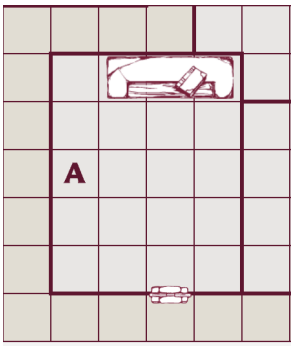
NOTE: If you want to place all heroes at specific points, you must use four separate ON_START Events, associated with separate Markers.

ON_SEARCH_TREASURES

This event is triggered when a hero performs a treasure search. If the marker is in a room, it is executed when a search is carried out in that room; if it is in a corridor, it is executed when a search is carried out and the marker is in line of sight with the hero carrying out the search.
NOTE: The REPEAT option can be specified for this Event.

- Example 1: **{[[(A), ON_SEARCH_TREASURES:REPEAT]] You only find rusted** weapons}
- Example 2: **{[[(A), ON_SEARCH_TREASURES]] Hidden between two dusty books in the bookshelf, you find a key. Write 'rusty key' on your** sheet}

With this example, the first search carried out in the room with marker A, instead of the normal screen that appears with the treasure search (which asks you to draw a card), the text 'Just in front of the fireplace, you find a key. Write 'rusty key' on your sheet'. Note how this allows you to customise the message with respect to the context (in this case the fact that there is a bookcase in the room).



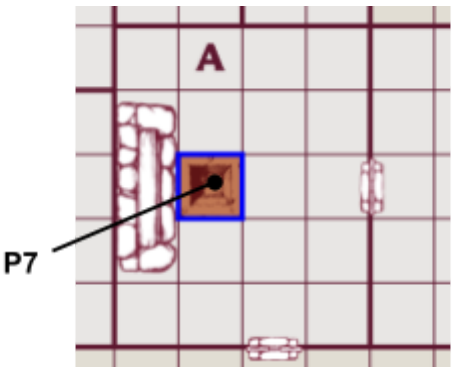
NOTE: According to the rules, searching in corridors would not be allowed. This is however possible in hQuestMaster, to give more elasticity to the system. If a Quest requires a variation from the official rules, it would be best to specify this at the beginning of the Quest. For example if a Key Event requires searching for treasures in a corridor it would be important to specify at the beginning that this is possible ("Heroes may search corridors"). Otherwise one might get stuck in the Quest without knowing why. In general it is useful to assume that players will use the official rules.

ON_SEARCH_TRAPS

This event is triggered at the first Trap search made by any hero. If the marker is in a Room, it is executed when a search is carried out in that room; if it is in a Corridor, it is executed when a search is carried out and the marker is in line of sight with the hero carrying out the search.
NOTE: The REPEAT option can be specified for this Event.

- ATTENTION: Using the ON_SEARCH_TRAPS event alone does not automatically make the trap visible. To do this, a SHOW action must be added to the box with the trap (see Example 2 below).
    - Example 1: **{[[(A), ON_SEARCH_TRAPS:REPEAT]] You don't find anything in the room!**}
    - Example 2: **{[[(A), ON_SEARCH_TRAPS]] Just in front of the fireplace, the floor seems unstable [[SHOW(P7)]]**}

In Example 1 above, with the trap search only a customised message 'You don't find anything dangerous in the room}' appears.
In Example 2, on the other hand, after searching, the message appears and a question mark appears in the P7 box, indicating that there is a trap that can be defused.



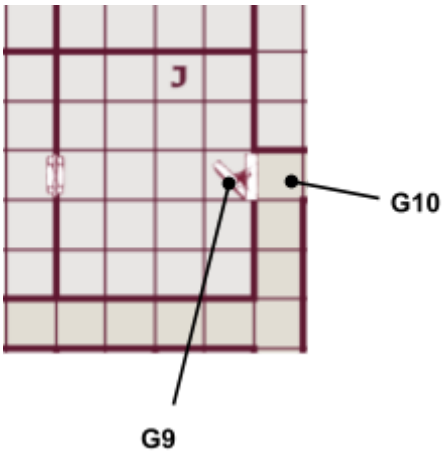
ON_SEARCH_SECRETDOORS

This Event is triggered at the first Secret Door search made by any hero. If the marker is in a Room, it is executed when a search is carried out in that room; if it is in a Corridor, it is executed when a search is carried out and the marker is in line of sight with the hero carrying out the search.
NOTE: The REPEAT option can be specified for this Event.

ATTENTION: If there is this event, the normal search for secret doors is not performed, and therefore the presence of any secret doors is not automatically displayed.

- Example 1: **{[[(J), ON_SEARCH_SECRETDOORS:REPEAT]] You don't find any secret door but it seems that there was an entrance in a wall in the past, now covered with** rocks}
- Example 2: **{[[(J), ON_SEARCH_SECRETDOORS]] You found a passage hidden between two tiles in the wall [[SHOW(G9)]]**}

In Example 1, when searching for secret ports, only a message is shown.
Example 2 instead combines the ON_SEARCH_SECRETDOORS event with the SHOW action to show the message and at the same time reveal the secret door in the box at co-ordinates G9. Note that each hidden door refers to two boxes (the two connected by the door). In Example 2 it would therefore also have been correct and equivalent to put SHOW(G10) instead of SHOW(G9).

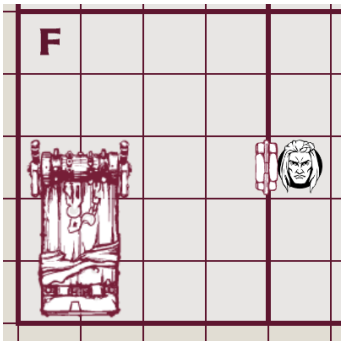


ON_ENTER_ROOM

This event is triggered when a hero enters the room where the Marker is present. Normally it is only triggered the first time one of the heroes enters the room, if the **:REPEAT** option is specified, the event will be triggered every time a hero enters the room

- Example 1: **{[[(F), ON_ENTER_ROOM]] The corpse of an elf lays on the torture rack}**
- Example 2: **{[[(F), ON_ENTER_ROOM:REPEAT]] Laying on the torture rack, you finally found the elf you were looking for. Unfortunately, there is no doubt he is already dead...**}

With this command, entering the room with the torture table will display the specified message. Whereas in Example 1 the message is only shown the first time, in Example 2 the message is repeated each time a hero enters the room.

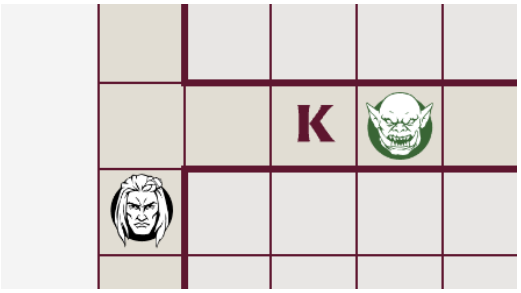


ON_SHOW

This Event is triggered when the cell where the marker is present becomes visible (i.e. the cell becomes from dark to visible).
NOTE: ON_SHOW **cannot** be changed with REPEAT

- Example 1 : **{[[(A), ON_SHOW:REPEAT]] You find a patrolling orc!**}
- Example 2 : **{[[(A), ON_SHOW]] You find a patrolling orc!**}

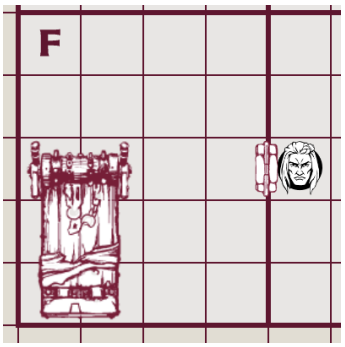
In this example, as soon as marker K is within the barbarian's line of sight, the message "You find a patrolling orc!" will appear on the screen



ON_OPEN

This event is triggered when a hero opens a door or secret door at the marker's location.
NOTE: for ON_OPEN **cannot** be changed with REPEAT

- Example: **{[[(F), ON_OPEN]] The door is rusty and you have to use force to open it}.**



NOTE: When planning ON_OPEN-related actions, remember that there are different ways to enter a room, other than by opening a specific door (e.g., by using another door, or by using the "Cross the Rock" spell). Entering the room without going through the door would not trigger the ON_OPEN event. In general it is suggested to use ON_OPEN only for Actions or messages that refer to the door (and not to the room). If the action is to take place at the entrance of the room it is better to use ON_ENTER_ROOM or, ON_SHOW, These Events are in fact triggered regardless of how you enter the room.

MONSTER_ON

This Event is always triggered automatically at the start of the game and is used to specify details of a monster already positioned on the map created with hQuestBuilder and whose co-ordinates are specified as the first of the arguments.

The mandatory parameter is the box to which the action refers and must be specified as the first argument.

- **MONSTER_ON(**monster_coordinates**, ...)**

Optional **Attributes**, and they are:

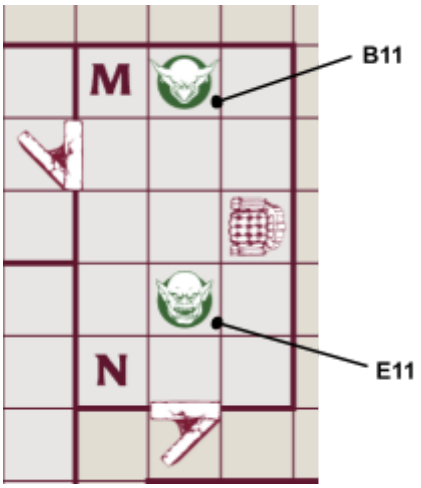
- NAME: The monster's proper name
- **MOV**: The number of cells you can move each turn
- **ATK**: Monster Attack
- DEF: Monster defense
- BODY: Monster Body Points
- **MIND**: Monster Mind Points
- **TYPE**: Currently the only allowable value is **PASSIVE**, which indicates that the monster is a passive type and will not attack heroes. If omitted the normal monster AI will be used.
- **SPELLS**: The spells the monster can cast, must be a semicolon-separated list (;) of the names of the spells to be cast.
  These are the permitted magics:

- ***summonUndead***
- ***summonOrc***
- ***tempest***
- ***command***
- ***fear***
- ***sleep***
- ***firestorm***
- ***ballOfFlame***
- ***escape[cell] :*** This is the only spell that has a parameter, the cell value indicated in square brackets. This cell is the one where the monster will teleport to when the **escape** magic is cast.

Example: **....,SPELLS=summonOrc;fear;escape[C12];sleep,....**

- Example 1: **{[[(M), MONSTER_ON(B12, NAME=Greep, MOV=1, ATK=1, DEF=1, BODY=1, MIND=2, TYPE=PASSIVE)]]**}
- Example 2: **{[[(N), MONSTER_ON(E11, NAME=Orc Mage,MOV=4, ATK=1, DEF=3, BODY=3, MIND=6, SPELLS=summonOrc;fear;escape[C12];sleep;ballOfFlame)]]**}

In the two examples above, attributes are specified for the two monsters in the room. Note how the two events are associated with markers (M, and N) whose position is not relevant to that of the monster. Event M, specifies the goblin's attributes in box B11. The name Greep will appear, it will have the entered attributes (move 1, attack 1, defense 1) and will be as a Passive type, i.e. it will neither move nor attack. Event N, specifies the attributes of the orc in box E11, whose name will be 'Orc Mage', will have the characteristics entered (movement 4, attack 1, etc.). It will also have three spells: summonOrc, Fear, Escape (which points to box C12 when used), Sleep *and* BallofFlame).

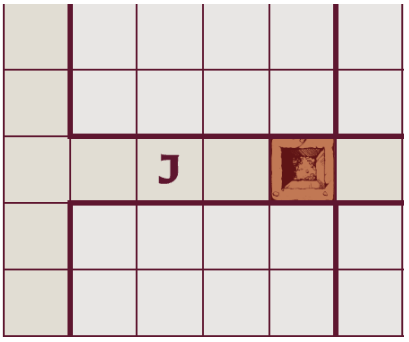


ON_STEP

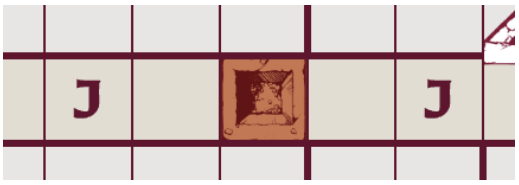
This Event is triggered when a hero takes a step into the specified cell.
NOTE: The REPEAT option can be specified for this Event.

- Example 1: **{[[(J), ON_STEP:REPEAT]] You hear a strange noise** }
- Example 2: **{[[(J), ON_STEP]] The floor of this corridor seems unstable** }

In the example, as soon as a hero takes a step, the text 'The floor seems unstable' will appear in the J box. As with any other Event, ON_STEP can also be combined with other actions.



NOTE: with ON_STEP it may be useful to use the marker several times, in various positions. In this case, the Event is triggered in any of the boxes with that specific Marker.

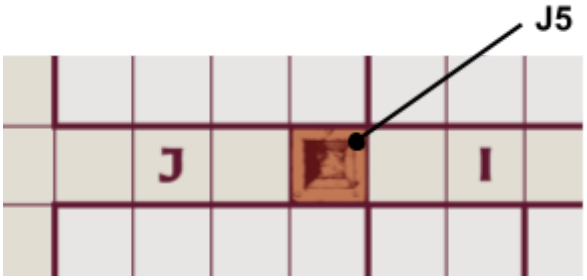


ON_TRAP_TRIGGER

This Event is triggered when a trap is activated in the specified cell
NOTE: This event is repeated each time the trap is activated (even if the REPEAT option is not specified).

- Mandatory parameters: (coordinate_cell)
- Example: **{[[(I), ON_TRAP_TRIGGER(J5)]] The pit is full of poisonous snakes! after resolving the pit normally, roll 2 battle dice and lose an additional BP for each skull** rolled}

In this Example, thanks to the ON_TRAP_TRIGGER, a different trap was created, modifying the displayed text which will be "The pit is full of poisonous snakes..".

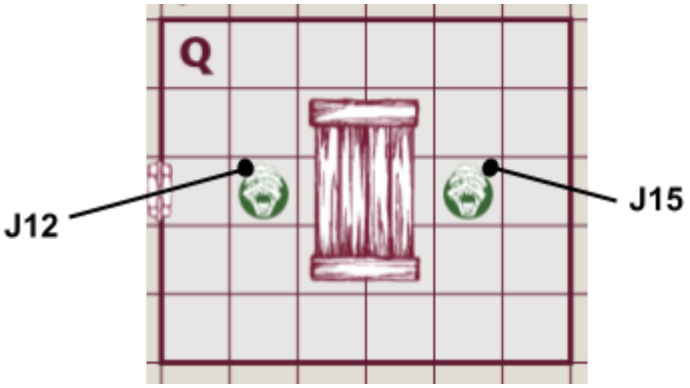


ON_DEATH

This Event is triggered when a monster (originally placed in the specified cell) dies. If several cells are indicated, it is only triggered when all those monsters have died.

- Parameters: (coordinate_cell1, coordinate_cell2, ...)
- Example: **{[[(Q), ON_DEATH(J12, J15)]] You've killed your targets, now you can escape the stairs!** }

In this example, the ON_DEATH event is associated with the killing of the two monsters starting at positions J12 and J15 (the two mummies in the example below). When both mummies are defeated the message "You've killed your targets, now you can escape the stairs!" will appear on the screen.



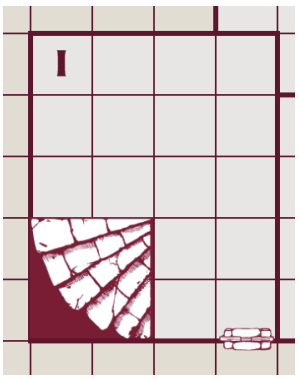
NOTE: This Event only works with monsters. If you want an Event to trigger on the death of a single hero use ON_SINGLE_HERO_DEATH, or if you want it to trigger on the death of all heroes use ON_ALL_HERO_DEATH.

ON_END_FIRSTHERO

This Event is executed when the first hero finishes the quest.

- Example: **{[[(I), ON_END_FIRSTHERO]] You are the first to escape alive the maze! you earn 200 gold pieces and the eternal glory!**}

In this example we assume that the heroes started from squares other than the stairs and that the goal is to escape from the stairs. In this example the first hero to reach them will display the message "You are the first to escape alive the maze! you earn 200 gold pieces and the eternal glory!". (Note that by default the stairs are the way out of the dungeon).



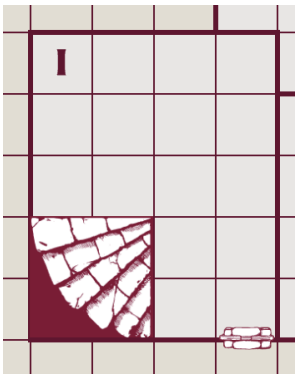
ON_END_ALLHEROES

This event is triggered when all heroes finish the quest.

- Example 1 : **{[[(I), ON_END_ALLHEROES]] You accomplished your mission. You can now return to the capital city and refer to the king what you discovered}**
- Example 2 : **{[[(I), ON_END_ALLHEROES]] [[IF: QUEST_OBJECTIVE_COMPLETED = 1]] You have completed your quest. You can now return to the capital city and refer to the king what you discovered [[QUEST_COMPLETE]] [[ELSE]] You escaped from the dungeon without the information you were looking for. Shame on you! [[QUEST_FAILED]] [[ENDIF]]}**

In Example 1 shown above, as soon as all heroes exit the stairs (by default, the dungeon exit) the message 'You accomplished your mission. You can now return to the capital city and refer to the king what you discovered'.

Example 2 uses a more complex action, namely checking that a certain objective has been completed. If it is, the text 'You accomplished your mission. You can now return to the capital city and refer to the king what you discovered' and the quest is completed. Otherwise, the text "You escaped from the dungeon without the information you were looking for. Shame on you!", after which the Quest will be failed. For the use of IF, QUEST_OBJECTIVE_COMPLETED, QUEST_COMPLETE, and QUEST_FAILED, see later in the List of Actions.

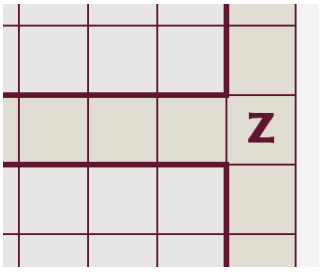


ON_ALL_HERO_DEATH

This event is triggered when all heroes are dead.

- Example 1: **{[[Z, ON_ALL_HERO_DEATH]] You did not save the princess. Your names will be forgotten}**

In the example note that the event was associated with the Z marker, placed in any square on the board. After the death of all heroes, the message "You did not save the princess. Your names will be forgotten'. NOTE: As with every QuestDown Event, a marker must be associated, in this case the marker can be placed in any square on the board.

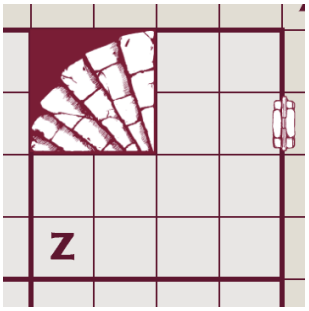


ON_SINGLE_HERO_DEATH

This Event is executed when any of the heroes dies. NOTE: This event repeats each time the trap is activated (even if the REPEAT option is not specified).

- Example 1: **{[[Z, ON_SINGLE_HERO_DEATH]] As the hero exhales the last breath, you can clearly see the soul, a pale and evanescent image, leaving the body}**

In this Example, after the death of each hero, the message "As the hero exhales the last breath, you can clearly see the soul, a pale and evanescent image, leaving the body" appears. Note that the example was associated with the Z marker near the entrance, but for ON_SINGLE_HERO_DEATH the marker can be anywhere on the board.

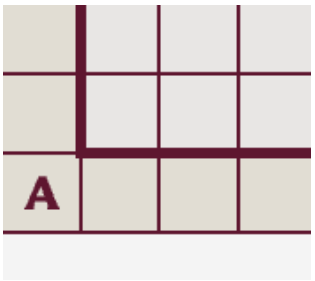


ON_NEW_TURN

This Event is executed at the beginning of each new heroes' turn (thus at the end of each Zargon turn)

- Example 1: **{[[A, ON_NEW_TURN]] You can feel the poison still flowing in your veins. Each hero rolls a combat dice, with a black shield, he/she loses one BP}.**

In this example, the words "You can feel the poison ... " appear at the start of each of the heroes' turns. NOTE: for ON_NEW_TURN, the position of the marker is not relevant, and is in this case placed in a corner of the map (any other box would have been fine).



HIDE_ON

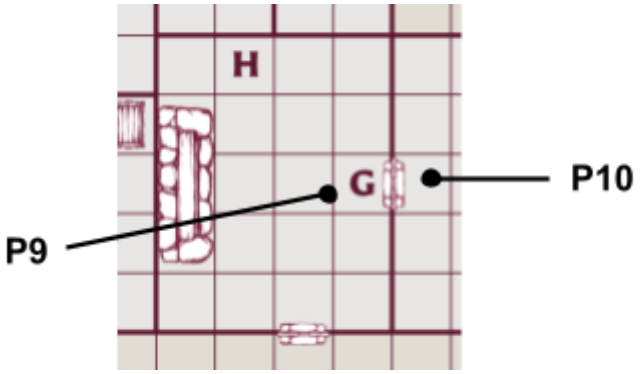
This Event is automatically triggered at the start of the game. It serves to indicate that at the start of the Quest the element (furniture item or door) positioned at the indicated cell is hidden, it will then be possible to make it visible with the **[[SHOW(cell_coordinates)]]** action.

- Parameters: [coordinate_cell].

- Example 1: {**[[H, HIDE_ON(P9)]]}**
- Example 2: {**[[H, HIDE_ON(P9)]]}**

{[[G, ON_STEP]] Upon stepping onto this point, you instantly notice the floor is somehow moving. Immediately after a door materialises in front of you, replacing what was a solid wall just moments ago [[SHOW(P9)]]}

In Example 1, we simply show how to use HIDE_ON. With this command, the door between boxes P9 and P10 is hidden (note that you only need to select one of the 2). The event is associated with the marker H, whose position is irrelevant (it could even be in another room).
In Example 2, we show the usefulness of HIDE_ON, combined with another Event, in this case ON_STEP.. Particularly with these two events the following would happen: at the entrance of the room the door between P9 and P10 would not be visible, but if a hero would pass over the box with the G-marker during movement, the ON_STEP event would be triggered. This event makes a message appear and also would make the door appear via the Action [[SHOW(P9)]].

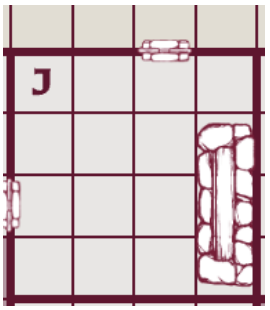


Inserting several Events for the same automation

With QuestDown it is also possible to have several Events for the same automation. For example, it might be possible to trigger a special message, either searching for treasures or searching for traps. In this case, Events must be written in succession separated by a semicolon (;).

For example:

- **{[[(H), ON_SEARCH_SECRETDOORS; ON_SEARCH_TREASURES; ON_SEARCH_TRAPS)]] you don't find anything but staying too long in this room you feel nauseated, you lose 1 BP}**



List of Actions

Each Event, by default, is associated with the appearance of text on the screen. **Actions**, on the other hand, are commands that allow various consequences to occur. Each Event may be associated with several Actions. By default, each Action is specified with this syntax **[[action]]**. Remember that **Actions must always be associated with an event according to the pattern {[[(marker), Event]] text or [[Action1]] [[Action2]]}.**

Some actions may require specific parameters (e.g. they are associated with specific cells), and some actions may also require additional associated commands.

Important Note on Actions

Below are examples of syntax on actions. it is important to remember that these must always be placed within an Event. So that it is the Event that causes the action to be executed.

Examples with specific Events will be given below for each action, but it is important to remember that one or more actions can be combined with any Event.

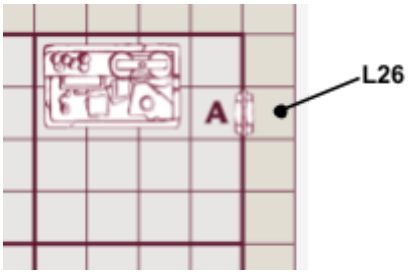
[[ASK: question]] ... [[ELSE]] ... [[END]]

This Action shows a question to the user and waits for an answer. In case of a yes answer (i.e., "Yes"), the next text is shown and/or actions up to the **[[ELSE]]** or **[[END]]** block are executed. If a negative answer (i.e., "No") is given, the next text is shown and/or the actions present between the **[[ELSE]]** and [[**END]] block are executed.** The **[[ELSE]] block** is optional.

- Example 1: ***{[[(A), ON_SEARCH_TREASURES]] [[ASK: Do you want to put your hand into a mysterious bag?]] You found a gem worth 100 gold pieces! [[END]]}***
- Example 2: ***{[[(A), ON_OPEN:REPEAT]] [[ASK: Do you have the key?]] [[OPEN(L26)]] [[ELSE]] [[LOCKED]] [[END]]}***

In Example 1*, the text **"Do you want to put your hand into a mysterious bag? If the answer is "Yes" the text "You found a gem worth 100 gold pieces!" will appear,** otherwise since there is no [[ELSE]] block, nothing will appear. Note that in this case, if the player answers "no", you will no longer be able to put your hand in the bag (because ON_SEARCH_TREASURES, does not repeat). NOTE: If you want the question to repeat until someone answers "Yes", you must use the IF Action (see below).*

In Example 2, this is an ASK which is triggered in response to an attempt to open a door next to Marker A (and is a repeated event). If a hero answers 'no', it will still be possible to try again. After the door is opened, it can no longer be opened (because it is already open) and therefore the ON_OPEN Event will no longer be triggered.

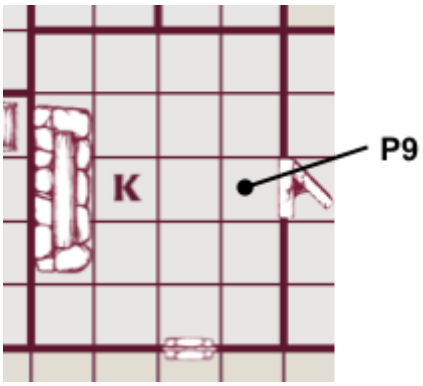


[[OPEN(cell_coordinates)]]

This Action opens a door, or secret door, in the specified cell.

- Parameters: **(cell_coordinates)**
- Example: **{[[(K), ON_STEP]] As soon as you step in front of the fireplace you hear a distinct click from the wall [[OPEN(Q14)]]}**

In this action, as soon as a hero walks over Marker K, the ON_STEP Event will be triggered. The text As soon as you step in front of the fireplace you hear a distinct click from the wall will then appear, and you will open the secret door on box P9.

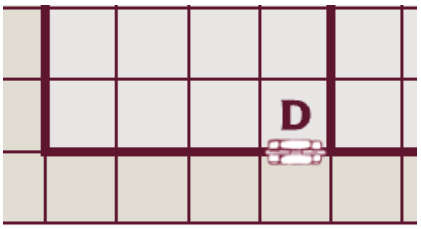


[[LOCKED]]

This action indicates that a door, or secret door, is locked and cannot be opened. Specifically, the message "The door is locked!" will appear.

- Example: **{[[(D), ON_OPEN]] [[LOCKED)]] }**

In this case, in response to the Door Open Event next to Marker D, the Default message will appear and the door cannot be opened.

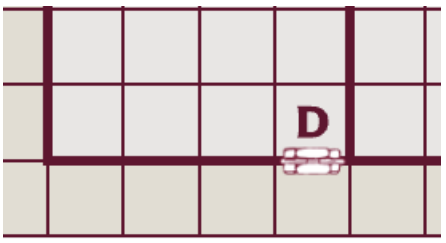


[[LOCKED: message]]

This action indicates that a door, or secret door, is locked and displays a customised message to the user.

- Example: {**[[(D), ON_OPEN]] [[LOCKED: this door is locked by magic!]] }**

In this example, the customised message 'this door is locked by magic!' will appear next to Marker D when the door is opened. The door also cannot be opened.

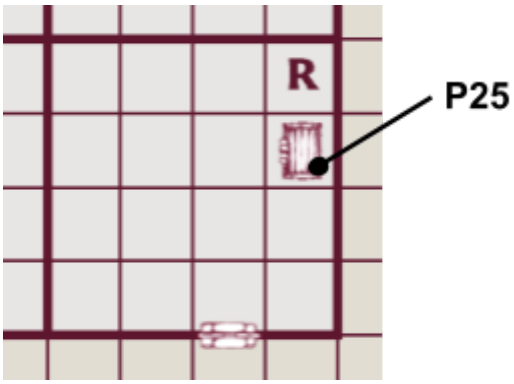


[[REMOVE(cell_coordinates)]]

This Action removes an element (furniture or door) from the cell specified by its name (i.e. from the co-ordinates detected by hQuestBuilder).

Parameters: **(coordinate_cell)**

- Example: **{[[R, ON_SEARCH_TREASURES]] As soon as you start searching in the room the chest disappears with a puff. You just lost your time. [[REMOVE(P25)]]**}



NOTE: The REMOVE action permanently removes a furniture item or door from the board. If you want the removal to be temporary, you must use the HIDE action.

[[ADD_NPC(cell_coordinates, attributes)]]

This action adds a non-player character (NPC) to the specified cell. This character will be shown with a default token in the map (a token with the letter X), and must be played (movement, attack, defense) by the players.

The mandatory parameter is:

- NAME: The proper name of the NPC.

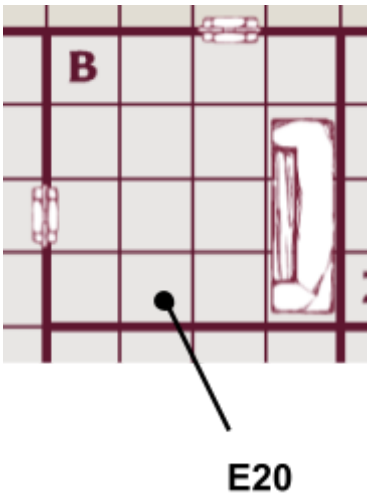
The optional parameter is:

- **ON_DEATH: This is** the event to be triggered in case of death of the NPC. Any event or message can be specified

    - Parameters: **(cell_coordinates, ...**)

    - **Example: {[[(B), ON_SHOW]] [[ADD_NPC(E20, NAME=Prince Bart, ON_DEATH=[[QUEST_FAILED]])]]}**

In this example, the moment you see box B (e.g. opening the door or entering the room, an NPC named Prince Bart will appear in box E20. If he is killed, the Quest will be ended as failed.



[[ADD_MONSTER(cell_coordinates, ...)]]

This action adds a monster to the specified cell. Instead of indicating the cell you can use the keyword NEAR which will place the monster in the cell closest to the hero whose turn it is.

- Parameters: **(cell_coordinates, ...)**

the mandatory Attributes, and they are:

- NAME: The Monster's proper name.
- KIND: The basic monster species: ***Skeleton, Zombie, Goblin, Orc, etc.***
- **TYPE**: Currently the only allowable value is **PASSIVE**, which indicates that the monster is a passive type and will not attack heroes. If omitted the normal monster AI will be used.
- **MOV**: The number of cells you can move each turn
- **ATK**: Monster Attack
- DEF: Monster defence
- BODY: Monster Body Points
- **MIND**: Monster Mind Points

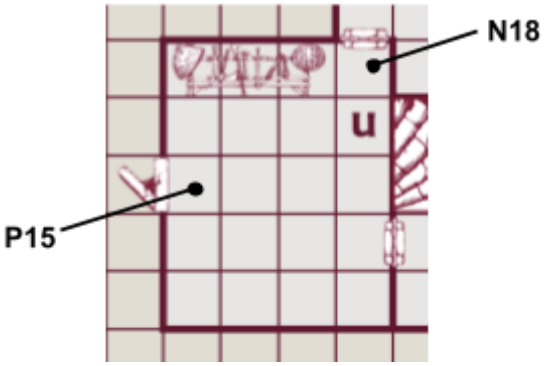
The optional Attributes are:

- **ON_DEATH**:: This is the event to be triggered in case of death of the NPC. Any Action or message can be specified
- **SPELLS**: The spells the monster can cast, must be a semicolon-separated list (;) of the names of the spells to be cast.
  These are the permitted magics:
  - ***summonUndead***
  - ***summonOrc***
  - ***tempest***
  - ***command***
  - ***fear***
  - ***sleep***
  - ***firestorm***
  - ***ballOfFlame***
  - ***escape[cell] :*** This is the only spell that has a parameter, the cell value indicated in square brackets. This cell is the one where the monster will teleport to when the **escape** magic is cast.

  Example: **....,SPELLS=summonOrc;fear;escape[C12];sleep,....**

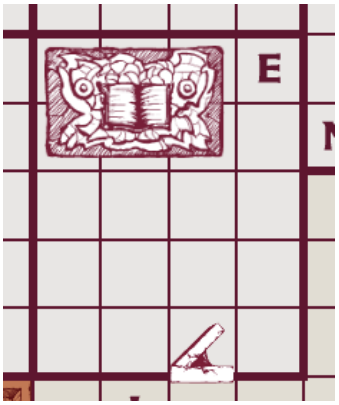
- Example 1: **{[[(U), ON_OPEN(P15)]] As you start opening the secret passage, the door bursts open and Grutgar, the powerful Orc captain, appears. You must kill it!. [[ADD_MONSTER(N18, NAME=Grutgar, KIND=Orc, MOV=10, ATK=5, DEF=5, BODY=5, MIND=2, ON_DEATH=[[QUEST_OBJECTIVE_COMPLETED]])]]**}

In this Example, as soon as a hero tries to open the secret door in box P15, first the message 'As you start opening the secret passage ...' will be shown. then a monster will appear in box N18, with the stats specified by the attributes.



- Example 2: **{[[(E), ON_SEARCH_TREASURES]] While you are looking for treasures in the room, you feel a sudden shiver. Few moments later, a horrific skeleton surrounded by a devilish aura appears just in front of you [[ADD_MONSTER(NEAR, NAME=Demon Skeleton, KIND=Skeleton, MOV=4, ATK=4, DEF=4, BODY=3, MIND=0, SPELLS=fear;ballOfFlame)]]}**

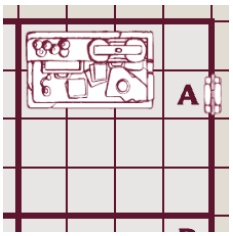
In this example, as soon as a hero tries to look for treasures in the room with marker E, the text 'While you are looking for treasures in the room ...' will appear. Then a monster with the specified attributes will be added. Note that since NEAR is specified as the location of the monster, wherever the hero searches, the monster will appear next to him/her.



[[QUEST_OBJECTIVE_COMPLETED]]

This action indicates that a quest objective has been completed, it is used in combination with [[QUEST_COMPLETE]] to determine that the Quest is completed. When specified with [[QUEST_COMPLETE]], the quest will only be evaluated as completed if the action [[QUEST_OBJECTIVE_COMPLETED]] has been activated first.

- Example: {**[[(A), ON_SEARCH_TREASURES]] Hidden in a drawer, you found the ancient tome you were looking for. Now you can leave the dungeon! [[QUEST_OBJECTIVE_COMPLETED]]}**



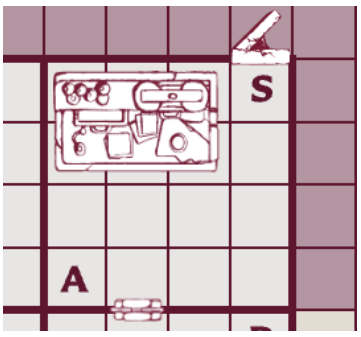
In this example, after searching for treasures in the room, it will be recorded that the quest objective is complete. At the end of the Quest using the Action [[QUEST_COMPLETE]], the Quest will only actually be complete if the objective was completed.

[[QUEST_COMPLETE]]

This Action indicates that the Quest has been completed. If this Action is not specified, the Quest ends independently when all heroes reach the stairs.

When the Quest ends with [[QUEST_COMPLETE]], it is automatically checked whether the objective of the Quest has been reached, i.e. the action [[QUEST_OBJECTIVE_COMPLETED]] has been used before**, if** yes, the player will be notified that the Quest has ended and has been completed; otherwise the player will be notified that the Quest has ended, but failed.

- Example**: {[[(S), ON_OPEN]] [[ASK: You found the secret passage that will lead you outside the dungeon. Do you want to exit?]] [[QUEST_COMPLETE]] [[ELSE]] [[LOCKED]] [[END]]}**

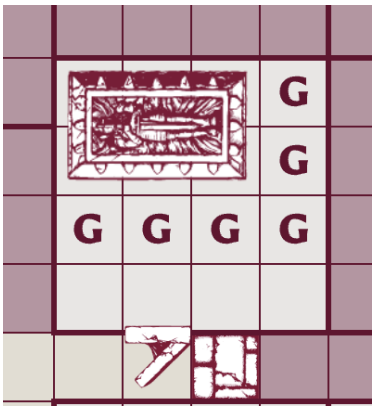


In this example once the secret door is found. Trying to open it will display the message 'You found the secret passage that ...'. Using the ASK action, you are asked whether you want to exit or not. If the answer is 'Yes', the Action [[QUEST_COMPLETE]] is activated and thus Quest ends. Read also the Example in QUEST_OBJECTIVE_COMPLETED. Combining the two Examples in QUEST_OBJECTIVE_COMPLETED and QUEST_COMPLETE, the Quest is only completed successfully if a search for treasures in the room was carried out first, otherwise it will be concluded with a failure.

[[QUEST_FAILED]]

Indicates that the mission has failed. After it is activated, the game on hQuestMaster will be terminated and a message indicating that the mission has failed will be displayed.

- Example: **{[[(G), ON_STEP]] You were told not to stay close to the cursed tomb. As you step closer, several ghosts start to circle around you and soon you realise that the ancient hex is now complete. Your Quest has failed! [[QUEST_FAILED]]}**



In this example as soon as a hero steps on one of the boxes with the G-marker, the message "You were told not to stay close ..." will appear on the screen and the Quest will be completed.
NOTE: That in this case a marker was used several times with the ON_STEP command. In this way only one of the boxes with the Marker needs to be stepped on and the Action will be activated.

[[SET(variable,value)]]

SET, is used to assign a value to a Variable. Together with the Action **[[IF ..]].** the SET Action allows for great versatility in QuestDown and allows for text or Actions to change in response to Events during the Quest.

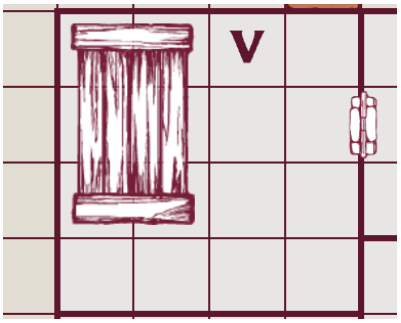
Variable names must be alphanumeric only, they may not contain spaces or other special characters. Assigned values may be numbers or text.

- [[SET(Key,1)]]
- [[SET(Key, found)]]
- [[SET(Goal1, complete)]]

Etc.

- Example 1: **{[[(V), ON_SEARCH_TREASURES]] After a long search, you found under the table a key covered in blood. [[SET(key,1)]]}**

In this example, after the search is carried out, the variable key is set to 1. Through the use of IF and GET (see later) it is then possible to create Events or Actions that will depend on whether the key has been found or not.



- Example 2: {**[[(V), ON_SEARCH_TREASURES]] After a long search, you found under the table a key covered in blood. [[SET(key, key_found)]]}**

Example 2 is identical to Example 1, however, instead of assigning key the value 1, the value key_found is assigned (this is irrelevant in terms of operation, but are two possibilities of using the SET Action).

- Example 3: **{[[(Z), ON_DEATH(C23)]] The Orc boss is dead! now you can exit the dungeon [[SET(boss_killed, 1)]]}**

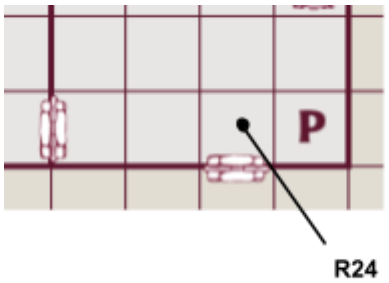
Using the code from Example 3 causes the variable boss_killed to be set to 1 after the Orc boss is killed. This element can be reused by other automations. (see examples with IF)

[[IF:logical_operation]] ... [[ELSEIF]] ... [[ENDIF]]

Actions associated with IFs are those that add versatility to QuestDown automations. In short, through the use of IF and other commands (according to the syntax specified below), you can create Events and Actions that will depend on certain Variables and may vary during the course of the game. For example, a door might turn out to be closed, but after pulling a lever, turn out to be open. Or a monster might only appear after exploring a certain room, etc. In practice this is resolved by one or more '*If .... then ...*' . For those unfamiliar with programming languages, it may not be straightforward to understand the principles involved in using IF and other commands. In such cases, it may be useful to see the examples below and use those as a starting point for developing your own automations.

Each Variable at the beginning has value 0.

The IF action works by first performing a **logical operation**. (see below). If the result of the logical operation is "True", the next text will be shown and/or actions up to the [[ELSEIF]] or [[ENDIF]] block will be executed, if it is "False", the next text will be shown and/or actions between the [[ELSEIF]] and [[ENDIF]] block will be executed. The [[ELSEIF]] block is optional:



- Example:**{[[(P), ON_OPEN(R24)]] [[IF: GET(lever) = 1]] You unlocked the door by pulling the lever before, so you can open the door! [[OPEN(S23)]] [[ELSEIF]] [[LOCKED]] [[ENDIF]]** }

In the example above, the text that appears depends on the Value of the Variable 'lever'. If the variable "lever" is equal to 1, the message "You unlocked the door by pulling the lever ..." appears. Otherwise, no special message appears and simply the door is locked (remember that each Variable has zero as its initial value). An Event with an IF use as just exemplified could have followed another previous command in which the lever variable was set to 1. For example:

{[[(K), ON_ENTER_ROOM:REPEAT]], [[ASK: Inside the room there is a lever. Do you want to pull it?]] You hear a strange sound from a nearby room [[SET(lever, 1]] [[END]]}

Note that in this case the lever can be pulled several times, in fact every time you enter the room. But once it has taken on a value of 1, each time it is pulled the same action is repeated and therefore nothing changes (the value is always set to 1).

NOTE: Those with programming experience should note that Variables do not need to be initialised before they can be used.

Logical operations

Logical operations involve the use of special keywords such as: **TURN**, **RND,QUEST_OBJECTIVE_COMPLETED.** In addition to these predefined keywords, it is also possible to define **Variables**, i.e. names that can be assigned a value and subsequently used in operations.

These keywords must be compared with a value using the logical operators =, <>, >, >=, <, <= with absolute values such as numbers or text. The operators allow checks that can result in 'True' or 'False'.

- **[[IF: GET(lever) = 1]]**

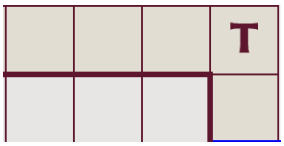
This example code checks that the Variable 'lever' has the value 1. If the value is 1 then the logical operation will result in 'True', in all other cases where lever does not have value = 1, it will result in 'False'.

- **[[IF: TURN > 3]]**

This example code checks that the Variable 'TURN' has a value greater than 3. For all values greater than 3, then the result of the logical operation will be 'True', in the case of 1 or 2, the result of the logical operation will be 'False'.

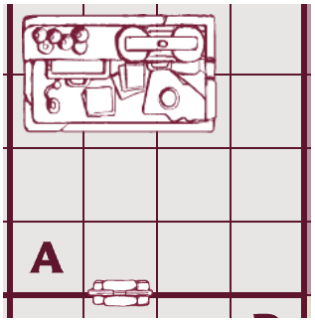
Below are some examples for various possibilities of using logical operations with the special keywords. For example the code

- TURN indicates the current turn of the game, however, so an IF can be created to verify that X turns have passed in order to perform an action or communicate a message to the players.
  - Example: **{[[(T), ON_NEW_TURN]] [IF: TURN > 20]] Too much time has passed, your quest has failed! [[QUEST_FAILED]] [[ENDIF]]}**



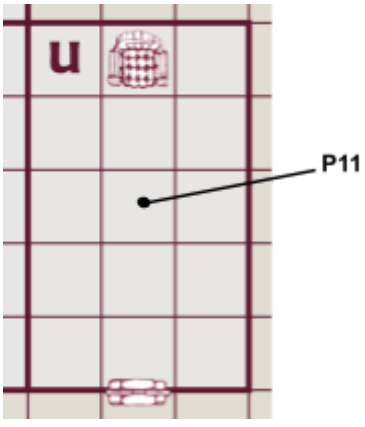
In this example at the start of each player's turn a check is made, if the Turn is greater than 10 then the message "Too much time has passed, .." appears and the Quest ends in failure. Note that as always for QuestDown, it is necessary to associate the Event with a Marker. In this case, as we use ON_NEW_TURN, it can be anywhere on the board.

- ***RND(DA,A)*** is used to generate a random number, it generates a number between the numbers. provided **DA** and **A**
  - Example: **{[[(A), ON_SEARCH_TREASURES]] [IF: RND(1,5) = 5]] You are lucky! You found 100 Gold coins! [[ELSEIF]] You don't find anything [[ENDIF]]}**



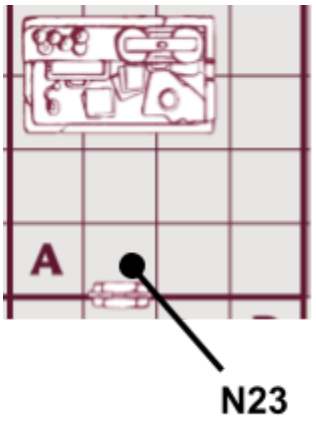
This Action with IF is activated when searching for treasures in the room with marker A. In this case there is a 1 in 5 chance of finding 100 gold coins. This is because a random number between 1 and 5 is drawn and if it is equal to 5, then the message 'You are lucky, you ...' appears. In all other cases the message 'You don't find anything' appears.

- ***QUEST_OBJECTIVE_COMPLETED*** is basically a default Variable which indicates whether the Quest objective has been reached. This operator is normally set equal to 0, unless the action **[[QUEST_OBJECTIVE_COMPLETED]]** was previously used, in which case it will have the value 1.
    - **Example:{[[(U), ON_DEATH(P11)]]**
      [[IF: QUEST_OBJECTIVE_COMPLETED = 1]] You have successfully killed Grutgar and found the scroll Now you can escape from the dungeon [[ELSEIF]] Oh no! You killed the evil wizard before finding the secret scroll that you were sent to find. You have failed! [[QUEST_FAILED]] [[ENDIF]])]}



In this example, after killing a monster that was in the P11 box, only if QUEST_OBJECTIVE_COMPLETED is equal to 1 will the text 'You have successfully killed ...' appear, otherwise the text 'Oh no! you killed the evil wizard ..' will appear and the Quest will fail.

- ***GET(name)*** is used to retrieve the value of a variable previously assigned with **[[SET(name,value)]].**
    - Example: **{[[(A), ON_OPEN(N23)]] [[IF: GET(key) = 1]] You have the Key so you can open this door! [[OPEN(S23)]] [[ELSEIF]] [[LOCKED]] [[ENDIF]**]



In this example, if the variable key has the value 1 then the text 'You have the Key so you can open this door' appears. Otherwise, the door is closed. Remember that each Variable has as its initial value 0. Note that programmed this way, the action will occur with any hero trying to open the door. However, more complex automation can be created with ASK, which can create a combination.

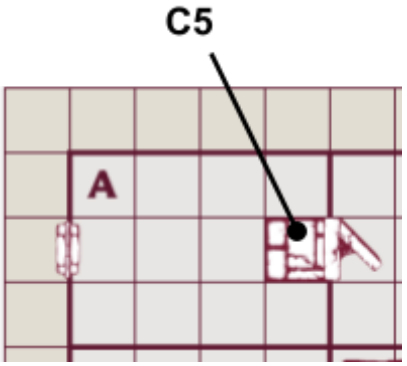
{[[(A), ON_OPEN(O23)]] [[IF: GET(key) = 1]] [[ASK: Do you have the key?]] The door is now open [[OPEN(S23)]] [[ELSE]] [[LOCKED]] [[END]] [[ELSEIF]] [[LOCKED]] [[ENDIF]]}

In this case what happens is that after you try to open the door, the phrase "Do you have the key?" will appear and only if the answer is yes, then the door will open. In all other cases the door remains closed.

[[HIDE(coordinate_cell)]]

This action hides a board element (furniture element or a normal or secret door or block) from the specified cell. This element can later be made visible again with the **[[SHOW(cell_coordinates)]]** action.

- Parameters: **(coordinate_cell**)
- Example**: {[[(A), ON_ENTER_ROOM:REPEAT]] As you enter the room you notice a lever in the middle of it [[IF: GET(block) <> 1]] [[ASK: do you want to pull the lever?]] you hear a noise and the block moves under the floor [[SET(block, 1)]] [[HIDE(C5)]] [[ELSE]] Nothing happens [[END]] [[ENDIF]]}**



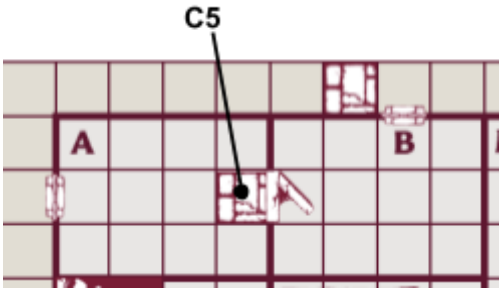
In this example, for each hero that enters the room the words As you enter the room you notice ..." will appear. Note that since there is a REPEAT option, the action will be repeated every time a hero enters the room. Afterwards if the variable "block" has a value other than 1 a question will be triggered:" Do you want to pull the lever?". If the answer is yes, then the variable block will be set to 1 and the block will be hidden via the HIDE action. With each further entry into the room (after the lever is pulled) the question will no longer appear.

[[SHOW(coordinate_cell)]]

This action allows an element (furniture or doors), previously hidden by the **[[HIDE(cell)]]** action or the **HIDE_ON** event, to be shown from the specified cell.

- Parameters: **(coordinate_cell)**
- Example:**{[[(B) ON_STEP]] you hear some noise from a nearby room [[SHOW(C5)]] [[SET(block, 0)]]}**

The example of SHOW is better understood when combined with the example shown earlier for HIDE. In this case, as soon as a hero steps on the box with marker B the message "you hear some noise from a nearby room" will appear and then the block in box C5 will be shown (assuming it was hidden before with HIDE or HIDE_ON).

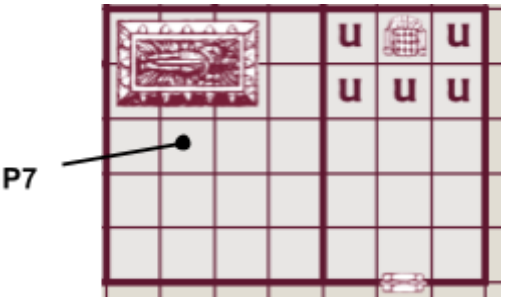


[[MOVE_HERO(destination_cell_coordinates)]]

This action teleports the currently active hero to the specified cell.

- Parameters: **(destination_cell_coordinates)**
- Example**:{[[(U), ON_STEP]] As you approach the throne you feel your body fading. Few moments after you find yourself in another room [[MOVE_HERO(P7)]]}**

In this example, as soon as a hero ends up in one of the squares with the U marker, the message 'As you approach the throne ...' will appear and the hero will be moved to square P7.



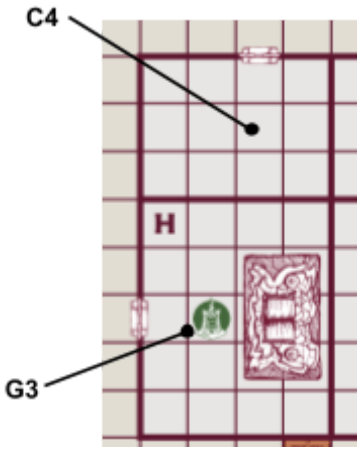
[[MOVE_MONSTER(monster_cell_coordinates,destination_cell_coordinates)]]

This action teleports the monster positioned at the cell with value **coordinate_cell_monster** to the specified cell **coordinate_cell_destination.**

Parameters: **(monster_cell_coordinates,destination_cell_coordinates)**

- Example:**{[[(H), ON_ENTER_ROOM] the evil wizard smiles and then disappears in a whirl of sparks [[MOVE_MONSTER(G3,C4)]]**}

In this example, as soon as one enters the room with Marker H, the message "The evil wizard smiles and then disappears . ' and the terror wizard is moved from square G3 to square C4.

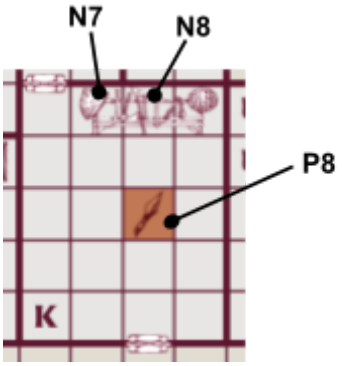


[[MOVE_OBJ(object_cell_coordinates,destination_cell_coordinates)]]

Teleports the object (furniture) positioned at cell coordinate_cell **to the** specified cell **coordinate_cell_destination.**

- Parameters: **(subject_cell_coordinates,destination_cell_coordinates**)
- Example {**[[(K), ON_TRAP_TRIGGER(P8)]] A trap springs, launching a spear (resolve the trap normally). Immediately afterwards a complex sets of gears activates moving the weapons rack just in front of you. [[MOVE_OBJ(N8,N7)]]**}

In this Example, when a hero goes into the P8 box, it triggers the ON_TRAP_TRIGGER(P8) Event. This event causes the text 'A trap springs, launching a spear ...' to appear and thus moving the rack from the N8 box to the N7 box.



This is a comprehensive overview of the events and actions supported by the QuestDown language. Use this documentation as a reference while developing your own interactive maps for hQuestMaster.

Recommendations/Suggestions

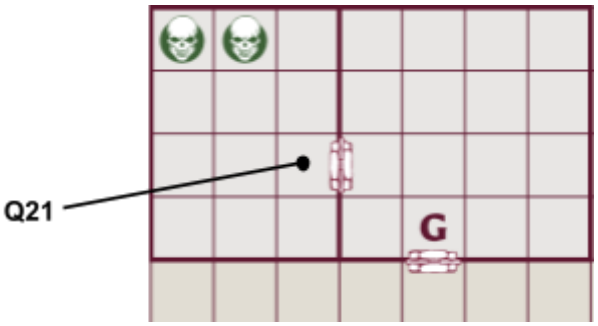
1) If you are not clear on how to specify code for QuestDown, in addition to seeing the examples in this document try the example adventures on HQM for inspiration. There are several working code examples in these adventures that you can reuse. Trying them out and seeing how they perform can be very useful. You can copy parts of the code and modify them to suit your Quest.
2) When you need to test your automations with QuestDown, you can use the combination **Ctrl+L** directly from hQuestBuilder to immediately start the QuestDown adventure.
3) Before sharing a Quest written with QuestDown, make sure (by debugging) that the automations behave as you expect in the Quest.
4) If you have to debug for automations, to speed up and save time, move the starting box of the heroes to the Event(s) you have to test.
5) Here are some common mistakes to pay particular attention to:
    a) One of the most common errors with QuestDown is the incorrect use of brackets (e.g. open and not closed brackets, use of wrong type of brackets, etc.). If a command does not work as it should, try to carefully check whether all parentheses are used correctly.
    b) Remember that each marker is generally associated with one Event, but can also be associated with more than one (see section Inserting Several Events for the Same Automation).
    c) When specifying an Event, remember the comma after the marker! (e.g. [[(M)**,** ON_OPEN]]).
6) If your QuestDown events alter the use of the basic rules, it is essential to make this explicit already from the introduction of the quest, or through special messages. For example, if an event is triggered by searching in a corridor, you should specify at the beginning of the adventure that it is also possible to search in corridors. In general, think of each adventure you share as an adventure that will be used by a person following the basic HeroQuest rules.
7) A powerful (yet simple) tool to enhance your Quest is the use of ON_ENTER_ROOM, accompanied by descriptive text that encapsulates the essence of the surroundings—details such as the sights, scents, and the reactions of the monsters within. A concise yet vivid portrayal helps in creating the atmosphere and heightening the immersive experience of your Quest.

Glossary

Attributes: these are those characteristics (mandatory or optional) that can be specified for monsters added via Events or Actions. For example ADD_MONSTER has a number of mandatory Attributes that define the monster's characteristics (Name, movement in boxes, body points, etc.) and as optional Attributes, any spells available to the monster.

Actions: **Actions** are commands that specify what will happen as a result of a specific Event. For example **[[OPEN]]** can be used to open a door or a secret door. In combination with an event, very special dynamic situations can be created. For example the command

{[[(G), ON_OPEN]] As soon as you open the door another door opens wide [[OPEN(Q21)]]}



The command indicates that as soon as you open the door in the box with the G marker, a door in the box with coordinates Q21 will open (*remember to see the coordinates of a box by pressing Q+D on hQuestBuilder at the same time*). If the box Q21 is the one with the door in the room to the left of the picture, this would be opened (and consequently the two skeletons would be placed).

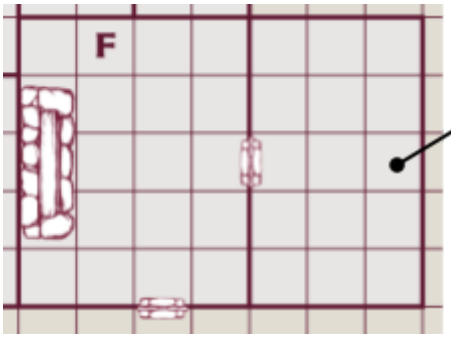
Events: **Events** are commands that specify *when* something specific will happen. For example, when a door is opened, when a trap is activated, or when a specific monster is defeated. An example of an event is **{[[(F), ON_ENTER_ROOM]] As soon as you enter the room you hear some rumbling noise coming from the nearby room [[ADD_MONSTER(K12, NAME=Orc, KIND=Orc, MOV=8, ATK=3, DEF=2, BODY=1, MIND=2]] }.**

Note the use of parentheses, which must be observed for the event to function correctly.

This event (as well as every other QuestDown event) is characterised by:

- A Marker (i.e. a letter in the map of hQuestBuilder that is associated with the event. In the example above it is the letter **F**
- the name of the Event (in the example above **ON_ENTER_ROOM**)
- i.e. which will happen when the event is triggered, which for each event, basically is simply the display of text. In the example above **As soon as you enter the room you hear some rumbling noise coming from the nearby room .** It is also possible to associate Actions to each Event, in this case **[[ADD_MONSTER(K12, NAME=Orc, KIND=Orc, MOV=8, ATK=3, DEF=2, BODY=1, MIND=2**]]

To summarise, when entering the room with the F marker, the text is first displayed and then an orc is added to the P12 box. The Event in this example shows QuestDown's versatility in handling automations.



s

Marker: a box in which a letter or number is marked in a Quest. In hQuestMaster it is used to be associated with an event**.**

NPC: non-player character, a character other than a monster, present in the Quest.

Parameter**:** within an Event or Action, additional arguments that help define automation are called 'Parameters'. The Lists of Events and Actions indicate the Parameters associated with each action. For example**,** the Event ON_OPEN contains a mandatory Parameter indicating the box with door or secret door to open

Example:

{[[(A), ON_OPEN(G16)]] The door is covered by human flesh...}

The mandatory parameter in this case is 'G16', which indicates that the event is triggered by opening the door in box G16.

Quest: a HeroQuest adventure, often implied to be developed with hQuestBuilder for use with hQuestMaster.

Variable: the name of an object in the QuestDown code that can be reused later and can take on different values. Variables are used to have automations that can change during the course of the game (e.g. a different text that appears if you have or do not have a certain artifact, found during the Quest). See the examples in SET, and IF.

All Variables have as initial value 0 (for those familiar with programming:, it is not necessary to 'initialise' them before using them in an Action with IF).

Credits

Documentation written by:

- **Ryuasd:** https://www.reddit.com/user/Ryuasd
- **Agorg:** https://www.reddit.com/user/Agorg2203

HELP & FAQ

To ask questions about QuestDown or for FAQs you can go to:

- **Reddit:** https://www.reddit.com/r/hQuestMaster/
- **GitHub:** https://github.com/Ryuasd/hQuestMaster/discussions

Index