Action Specials

Line Specials

This is a list of the Line Specials supported by Vavoom, they work in any of the games, feel free to add more information or examples to any of them.

ACS Specials

80:ACS_Execute (script, map, s_arg1, s_arg2, s_arg3)

- script: Script to execute
- map: Map which contains the script
- s_arg1: First argument passed to the script
- s_arg2: Second argument passed to the script
- s_arg3: Third argument passed to the script

Executes the specified script. A map value of zero indicates that the script is on the current map. If the script is on a different map, then the execution of the script will be delayed until the player enters the map that contains it. Only one copy of a script can be running at a time when started with this special.

If the specified script was previously executed but then suspended, then execution will begin at the point immediately after where it was suspended instead of starting over again at the beginning.

81:ACS_Suspend (script, map)

- script: Script to suspend
- map: Map which contains the script

Suspends execution of a script until an ACS_Execute or ACS_LockedExecute special starts it. If the specified script is not currently running, then it will be immediately suspended the next time it is run.

82:ACS_Terminate (script, map)

- script: Script to suspend
- map: Map which contains the script

Terminates execution of the specified script. You may not terminate scripts that were executed using the ACS ExecuteAlways special or ENTER scripts.

83:ACS_LockedExecute (script, map, s_arg1, s_arg2, lock)

- script: Script to execute
- map: Map which contains the script
- s_arg1: First argument passed to the script
- s_arg2: Second argument passed to the script
- lock: Required key, if any (see key types)

Executes the specified script if the player has the right key. A map value of zero indicates that the script is on the current map. If the script is on a different map, then the execution of the script will be delayed until the player enters the map that contains it. Only one copy of a script can be running at a time when started with this special.

If the specified script was previously executed but then suspended, then execution will begin at the point immediately after where it was suspended instead of starting over again at the beginning.

84:ACS_ExecuteWithResult (script, s_arg1, s_arg2, s_arg3)

- script: Script to execute
- s_arg1: First argument passed to the script
- s_arg2: Second argument passed to the script
- s_arg3: Third argument passed to the script

This is like ACS_ExecuteAlways, except the script is always run on the current map, and the return value is whatever the script sets with SetResultValue.

Example:

```
script 1 (void)
{
  print(d:ACS_ExecutewithResult (2)); // prints 667
}

script 2 (void)
{
  setresultvalue(667);
}
```

226:ACS_ExecuteAlways (script, map, s_arg1, s_arg2, s_arg3)

- script: Script to execute
- map: Map which contains the script
- s_arg1: First argument passed to the script
- s_arg2: Second argument passed to the script
- s_arg3: Third argument passed to the script

Like ACS_Execute, this special starts a script. However, it will allow multiple instance of a script to run simultaneously. The downside is that any scripts started with this special cannot be suspended or terminated with ACS_Suspend or ACS_Terminate. This is intended to be used from inside another script only, where you want to have one script do something at the same time as a "subtask" and it doesn't matter if there is more than one script running at once.

Ceiling Specials

Ceiling_Waggle(tag, amp, freq, offset, time)

- tag: Tag of affected sector
- amp: Amplitude of the waggle (in 1/8 of a unit)
- freq: Frequency of the waggle
- offset: Phase offset of the waggle
- time: How many tics the waggle lasts (0 tics means it will waggle forever)

"Waggles" the ceiling of the affected sectors in a sine wave.

40:Ceiling_LowerByValue (tag, speed, height)

- tag: Tag of affected sector
- speed: How quickly the ceiling moves
- height: Amount to lower ceiling by

Lowers a tagged sector's ceiling by height units. If tag is 0, then the sector on the line's back side is used.

41:Ceiling_RaiseByValue (tag, speed, height)

- tag: Tag of affected sector
- speed: How quickly the ceiling moves
- height: Amount to raise ceiling by

Raises a tagged sector's ceiling by height units. If tag is 0, then the sector on the line's back side is used.

42:Ceiling_CrushAndRaise (tag, speed, crush)

- tag: Tag of affected sector
- speed: How quickly the ceiling moves
- crush: Amount of damage to apply

Lowers and raises the ceiling of the affected sectors continually, applying crushing damage to anything underneath it. The ceiling will rise at half the speed at which it lowers. If tag is 0, then the sector on the line's back side is used.

43:Ceiling_LowerAndCrush (tag, speed, crush)

- tag: Tag of affected sector
- speed: How quickly the ceiling moves
- crush: Amount of damage to apply

Lowers the ceiling of affected sectors and applies crushing damage to anything under it. If tag is 0, then the sector on the line's back side is used.

44:Ceiling_CrushStop (tag)

- tag: Tag of affected sector

Stops a crushing ceiling.

45:Ceiling_CrushRaiseAndStay (tag, speed, crush)

- tag: Tag of affected sector
- speed: How quickly the ceiling moves
- crush: Amount of damage to apply

Lowers the ceiling of the affected sector and applies crushing damage to anything underneath it, then raises the ceiling back up to its original height. If tag is 0, then the sector on the line's back side is used. The up speed for this special is always half the down speed. If you need more control over the way the crusher moves, you have to use Ceiling_CrushRaiseAndStayA.

69:Ceiling_MoveToValueTimes8 (tag, speed, height, neg)

- tag: Tag of affected sector
- speed: How quickly the ceiling moves
- height: Absolute height of the move
- neg: Whether or not the destination height is negative

Moves the ceiling of affected sectors to an absolute height of (height * 8) units. If the destination height is negative, then neg should be 1, otherwise it should be 0 for a positive height. If tag is 0, then the sector on the line's back side is used.

192:Ceiling_LowerToHighestFloor (tag, speed)

- tag: Tag of affected sector
- speed: How quickly the ceiling moves

Lowers a ceiling to the height of the highest floor sorrounding it. If tag is 0, then the sector on the line's back side is used.

193:Ceiling_LowerInstant (tag, arg1, height)

- tag: Tag of affected sector
- arg1: Unused

- height: Amount to lower ceiling by

Instantly lowers a sector’s ceiling by (height * 8) units. If tag is 0, then the sector on the line’s back side is used.

194:Ceiling_RaiseInstant (tag, arg1, height)

- tag: Tag of affected sector
- arg1: Unused
- height: Amount to lower ceiling by

Instantly raises a sector’s ceiling by (height * 8) units. If tag is 0, then the sector on the line’s back side is used.

195:Ceiling_CrushRaiseAndStayA (tag, dspeed, uspeed, crush)

- tag: Tag of affected sector
- dspeed: How quickly the ceiling moves down
- uspeed: How quickly the ceiling moves back up
- crush: Amount of damage to apply

Lowers the ceiling of the affected and applies crushing damage to anything underneath it, then raises the ceiling back up to its original height. If tag is 0, then the sector on the line’s back side is used.

196:Ceiling_CrushAndRaiseA (tag, dspeed, uspeed, crush)

- tag: Tag of affected sector
- dspeed: How quickly the ceiling moves down
- uspeed: How quickly the ceiling moves back up
- crush: Amount of damage to apply

Lowers and raises the ceiling of the affected sectors continually, applying crushing damage to anything underneath it. If tag is 0, then the sector on the line’s back side is used.

197:Ceiling_CrushAndRaiseSilentA (tag, dspeed, uspeed, crush)

- tag: Tag of affected sector
- dspeed: How quickly the ceiling moves down
- uspeed: How quickly the ceiling moves back up
- crush: Amount of damage to apply

Lowers and raises the ceiling of the affected sectors continually, applying crushing damage to anything underneath it. If tag is 0, then the sector on the line’s back side is used. Crushers started with this special will only make noise when they reach the top or bottom of their strokes.

198:Ceiling_RaiseByValueTimes8 (tag, speed, height)

- tag: Tag of affected sector
- speed: How quickly the ceiling moves
- height: Amount to raise ceiling by

Raises a tagged sector’s ceiling by (height * 8) units. If tag is 0, then the sector on the line’s back side is used.

252:Ceiling_RaiseToNearest (tag, speed)

- tag: Tag of affected sector
- speed: How quickly the ceiling moves

Raises a ceiling to the height of the nearest surrounding ceiling. If tag is 0, then the sector on the line’s back side is used.

199:Ceiling_LowerByValueTimes8 (tag, speed, height)

- tag: Tag of affected sector
- speed: How quickly the ceiling moves
- height: Amount to lower ceiling by

Lowers a tagged sector's ceiling by (height * 8) units. If tag is 0, then the sector on the line's back side is used.

253:Ceiling_LowerToLowest (tag, speed)

- tag: Tag of affected sector
- speed: How quickly the ceiling moves

Lowers a ceiling to the height of the lowest surrounding floor. If tag is 0, then the sector on the line's back side is used.

254:Ceiling_LowerToFloor (tag, speed)

- tag: Tag of affected sector
- speed: How quickly the ceiling moves

Lowers a ceiling to the height of the floor underneath it. If tag is 0, then the sector on the line's back side is used.

255:Ceiling_CrushRaiseAndStaySilA (tag, dspeed, uspeed, crush)

- tag: Tag of affected sector
- dspeed: How quickly the ceiling moves down
- uspeed: How quickly the ceiling moves back up
- crush: Amount of damage to apply

Lowers the ceiling of the affected and applies crushing damage to anything underneath it, then raises the ceiling back up to its original height. If tag is 0, then the sector on the line's back side is used. Crushers started with this special will only make noise at the top and bottom of their strokes.

Door Specials

10:Door_Close (tag, speed, lighttag)

- tag: Tag of affected sector
- speed: How quickly the door closes
- lighttag: Tag of sector to perform a gradual lighting effect in

Lowers the ceiling of all affected sectors to the floor. If tag is 0, then the sector on the line's back side is used.

If lighttag is non-zero a gradual lighting effect is done in the tagged sectors. The light is gradually changed between the darkest neighboring sector when the door is fully closed and the brightest neighboring sector when the door is fully open.

11:Door_Open (tag, speed, lighttag)

- tag: Tag of affected sector
- speed: How quickly the door opens
- lighttag: Tag of sector to perform a gradual lighting effect in

Raises the ceiling of all affected sectors to four units below the lowest surrounding ceiling. If tag is 0, then the sector on the line's back side is used.

If lighttag is non-zero a gradual lighting effect is done in the tagged sectors. The light is gradually changed between the darkest neighboring sector when the door is fully closed and the brightest neighboring sector when the door is fully open.

12:Door_Raise (tag, speed, delay, lighttag)

- tag: Tag of affected sector
- speed: How quickly the door moves
- delay: Tics until door closes
- lighttag: Tag of sector to perform a gradual lighting effect in

Raises the ceiling of all affected sectors to four units below the lowest surrounding ceiling. After the door is opened, it will be closed again after delay tics. If tag is 0, then the sector on the line’s back side is used.

If lighttag is non-zero a gradual lighting effect is done in the tagged sectors. The light is gradually changed between the darkest neighboring sector when the door is fully closed and the brightest neighboring sector when the door is fully open.

13:Door_LockedRaise (tag, speed, delay, lock, lighttag)

- tag: Tag of affected sector
- speed: How quickly the door moves
- delay: Tics until door closes (0 if never)
- lock: Required key (see key types)
- lighttag: Tag of sector to perform a gradual lighting effect in

Raises the ceiling of all affected sectors to four units below the lowest surrounding ceiling if the player has the proper key. After the door is opened, it will be closed again after delay tics. If tag is 0, then the sector on the line’s back side is used.

If lighttag is non-zero a gradual lighting effect is done in the tagged sectors. The light is gradually changed between the darkest neighboring sector when the door is fully closed and the brightest neighboring sector when the door is fully open.

Floor Specials

20:Floor_LowerByValue (tag, speed, height)

- tag: Tag of affected sector
- speed: How quickly the floor moves
- height: Amount to lower floor by

Lowers a tagged sector’s floor by height units. If tag is 0, then the sector on the line’s back side is used.

21:Floor_LowerToLowest (tag, speed)

- tag: Tag of affected sector
- speed: How quickly the floor moves

Lowers a tagged sector’s floor to the height of the lowest surrounding floor. If tag is 0, then the sector on the line’s back side is used.

22:Floor_LowerToNearest (tag, speed)

- tag: Tag of affected sector
- speed: How quickly the floor moves

Lowers a tagged sector’s floor to the height of the highest surrounding floor. If tag is 0, then the sector on the line’s back side is used.

23:Floor_RaiseByValue (tag, speed, height)

- tag: Tag of affected sector
- speed: How quickly the floor moves
- height: Amount to raise floor by

Raises a tagged sector's floor by height units. If tag is 0, then the sector on the line's back side is used.

24:Floor_RaiseToHighest (tag, speed)

- tag: Tag of affected sector
- speed: How quickly the floor moves

Raises a tagged sector's floor to the height of the highest surrounding floor. If tag is 0, then the sector on the line's back side is used.

25:Floor_RaiseToNearest (tag, speed)

- tag: Tag of affected sector
- speed: How quickly the floor moves

Raises a tagged sector's floor to the height of the next higher surrounding floor. If tag is 0, then the sector on the line's back side is used.

28:Floor_RaiseAndCrush (tag, speed, crush)

- tag: Tag of affected sector
- speed: How quickly the floor moves
- crush: Amount of damage to apply

Raises the floor to a height of 8 units below the ceiling and applies crushing damage to anything standing on it. If tag is 0, then the sector on the line's back side is used.

35:Floor_RaiseByValueTimes8 (tag, speed, height)

- tag: Tag of affected sector
- speed: How quickly the floor moves
- height: Amount to raise floor by

Raises a tagged sector's floor by (height * 8) units. If tag is 0, then the sector on the line's back side is used.

36:Floor_LowerByValueTimes8 (tag, speed, height)

- tag: Tag of affected sector
- speed: How quickly the floor moves
- height: Amount to lower floor by

Lowers a tagged sector's floor by (height * 8) units. If tag is 0, then the sector on the line's back side is used.

46:Floor_CrushStop (tag)

- tag: Tag of affected sector

Stops a crushing floor.

66:Floor_LowerInstant (tag, arg1, height)

- tag: Tag of affected sector
- arg1: Unused
- height: Height of move

Instantly lowers the floor of the affected sectors by (height * 8) units. If tag is 0, then the sector on the line's back side is used.

67:Floor_RaiseInstant (tag, arg1, height)

- tag: Tag of affected sector
- arg1: Unused
- height: Height of move

Instantly raises the floor of the affected sectors by (height * 8) units. If tag is 0, then the sector on the line's back side is used.

68:Floor_MoveToValueTimes8 (tag, speed, height, neg)

- tag: Tag of affected sector
- speed: How quickly the floor moves
- height: Absolute height of the move
- neg: Whether or not the destination height is negative

Moves the floor of affected sectors to an absolute height of (height * 8) units. If the destination height is negative, then neg should be 1, otherwise it should be 0 for a positive height. If tag is 0, then the sector on the line's back side is used.

138:Floor_Waggle (tag, amp, freq, offset, time)

- tag: Tag of affected sector
- amp: Amplitude of the waggle (in 1/8 of a unit)
- freq: Frequency of the waggle
- offset: Phase offset of the waggle
- time: How many tics the waggle lasts (0 tics means it will waggle forever)

"Waggles" the floor of the affected sectors in a sine wave.

235:Floor_TransferTrigger (tag)

- tag: Tag of affected sector

Transfers the floor picture and sector special from one sector to another using the trigger change model.

236:Floor_TransferNumeric (tag)

- tag: Tag of affected sector

Transfers the floor picture and sector special from one sector to another using the numeric change model.

238:Floor_RaiseToLowestCeiling (tag, speed)

- tag: Tag of affected sector
- speed: How quickly the floor moves

Raises a tagged sector's floor to the height of the lowest surrounding ceiling. If tag is 0, then the sector on the line's back side is used.

239:Floor_RaiseByValueTxTy (tag, speed, height)

- tag: Tag of affected sector
- speed: How quickly the floor moves
- height: Amount to raise floor by

Raises a tagged sector’s floor by height units and uses the trigger change model to give it a new picture and special. If tag is 0, then the sector on the line’s back side is used.

242:Floor_LowerToHighest (tag, speed, adjust)

- tag: Tag of affected sector
- speed: How quickly the floor moves
- adjust: Amount of difference from target height + 128

Lowers a tagged sector’s floor to the height of the highest surrounding floor + adjust – 128. So if you want the floor to lower to the height of the highest surrounding floor, use an adjust of 128. If you want it to lower to 8 units below the other floor, use an adjust of 120. Similar for other values of adjust. If tag is 0, then the sector on the line’s back side is used.

250:Floor_Donut (ptag, pspeed, sspeed)

- ptag: Tag of the pillar in the center of the donut
- pseed: How quickly to lower the pillar
- sspeed: How quickly to raise the surrounding sector’s floor

Performs a “donut” action on the specified sectors.

241:Floor_LowerToLowestTxTy (tag, speed)

- tag: Tag of affected sector
- speed: How quickly the floor moves

Lowers a tagged sector’s floor to the height of the lowest surrounding floor and uses the trigger change model to give it a new picture and special. If tag is 0, then the sector on the line’s back side is used.

241:Floor_LowerToLowestTxTy (tag, speed)

- tag: Tag of affected sector
- speed: How quickly the floor moves

Lowers a tagged sector’s floor to the height of the lowest surrounding floor and uses the trigger change model to give it a new picture and special. If tag is 0, then the sector on the line’s back side is used.

Light Specials

109:Light_ForceLightning

Makes lightning flash immediately (if the map also has lightning set in its MAPINFO).

110:Light_RaiseByValue (tag, value)

- tag: Tag of affected sector
- value: Amount to raise the light by

Increases the light level in a sector by value.

111:Light_LowerByValue (tag, value)

- tag: Tag of affected sector
- value: Amount to lower the light level by

Decreases the light level in a sector by value.

112:Light_ChangeToValue (tag, value)

- tag: Tag of affected sector
- value: New light level

Sets the light level in a sector to value.

113:Light_Fade (tag, value, tics)

- tag: Tag of affected sector
- value: New light level
- tics: How long the light takes to fade to the new level

Changes the light level in a sector gradually over a period of time until it reaches value.

114:Light_Glow (tag, upper, lower, tics)

- tag: Tag of affected sector
- upper: Upper light level
- lower: Lower light level
- tics: How long to change between two light levels

Fades the light level in a tagged sector between upper and lower for a duration determined by tics. If tics is specified as 0, the effect will be permanent.

115:Light_Flicker (tag, upper, lower)

- tag: Tag of affected sector
- upper: Upper light level
- lower: Lower light level

Switches the light level in a sector between upper and lower at random intervals between approximately 0.2 and 1.8 seconds.

116:Light_Strobe (tag, upper, lower, u-tics, l-tics)

- tag: Tag of affected sector
- upper: Upper light level
- lower: Lower light level
- u-tics: Time to stay at upper light level
- l-tics: Time to stay at lower light level

Switches the light level in a sector between upper and lower at the given intervals.

117:Light_Stop (tag)

Terminates lighting effect in tagged sector. The light level is left at whatever it was when the effect was halted.

232:Light_StrobeDoom (tag, u-tics, l-tics)

- tag: Tag of affected sector
- u-tics: Time to stay at upper light level
- l-tics: Time to stay at lower light level

This is the same as Light_Strobe except that upper is whatever the sector’s light level is right now, and lower is whatever the lowest light level in the surrounding sectors is. If the surrounding sectors are at the same light level as the affected sector, then lower is 0.

233:Light_MinNeighbor (tag)

- tag: Tag of affected sector

Sets the light level of a sector to match the lowest light level found in one of its neighboring sectors.

234:Light_MaxNeighbor (tag)

- tag: Tag of affected sector

Sets the light level of a sector to match the highest light level found in one of its neighboring sectors.

Pillar Specials

29:Pillar_Build (tag, speed, height)

- tag: Tag of affected sector
- speed: Speed of the build
- height: Height (relative to floor) where the floor and ceiling meet

Raises the floor of a sector and lowers its ceiling so that they meet. The speed argument controls the speed of whichever part of the pillar has to move further, and the other part will have its speed set so that it arrives at its destination at the same time. If height is 0, then the floor and ceiling will meet halfway between their original positions.

30:Pillar_Open (tag, speed, fdist, cdist)

- tag: Tag of affected sector
- speed: Speed of the open
- fdist: How far the floor should lower
- cdist: How far the ceiling should rise

Opens a pillar. If fdist is 0, then the pillar's floor will lower to the lowest surrounding floor. Similarly, if cdist is 0, then the pillar's ceiling will rise to the highest surrounding ceiling. The speed argument controls the speed of whichever part of the pillar has to move further, and the other part will have its speed set so that it arrives at its destination at the same time.

Note: This will only work on sectors that have equal ceiling and floor heights.

94:Pillar_BuildAndCrush (tag, speed, height, crush)

- tag: Tag of affected sector
- speed: Speed of the build
- height: Height (relative to floor) where the floor and ceiling meet
- crush: Amount of damage to apply

This is the same as Pillar_Build except that it will also apply crushing damage to anything blocking the pillar from closing.

Polyobject Specials

1:Polyobj_StartLine (po, mirror, sound)

- po: which polyobj is being defined
- mirror: polyobj that will mirror this one's movements
- sound: door sound sequence to play when this polyobj moves

Starts defining a polyobject.

2:Polyobj_RotateLeft (po, speed, angle)

- po: polyobj to rotate
- speed: how quickly the polyobj should rotate

- angle: byte angle to rotate the polyobj through

Rotates a polyobject left through the specified angle. If angle is 255, then the polyobject will rotate continuously and never stop.

3:Polyobj_RotateRight (po, speed, angle)

- po : polyobj to rotate
- speed: how quickly the polyobj should rotate
- angle: byte angle to rotate the polyobj through

Rotates a polyobject right through the specified angle. If angle is 255, then the polyobject will rotate continuously and never stop.

4:Polyobj_Move (po, speed, angle, dist)

- po: polyobj to move
- speed: how quickly the polyobj should move
- angle: direction the polyobj should move (this is a byte angle)
- dist: distance to move

Moves a polyobject.

5:Polyobj_ExplicitLine (po, order, mirror, sound)

- po: polyobj that is being defined
- order: rendering order of this line
- mirror: polyobj that will mirror the moves of this one
- sound: door sound sequence to play when this polyobj moves

Explicitly includes a line as part of a polyobject.

6:Polyobj_MoveTimes8 (po, speed, angle, dist)

- po: polyobj to move
- speed: how quickly the polyobj should move
- angle: direction the polyobj should move (this is a byte angle)
- dist: distance to move in units of 8

Moves a polyobject (dist * 8) units.

7:Polyobj_DoorSwing (po, speed, angle, delay)

- po: polyobj to move
- speed: how quickly to spin the polyobj
- angle: byte angle to rotate the polyobj through
- delay: delay in tics before returning to original orientation

Rotates a polyobject, waits, and then rotates it in the opposite direction until it has returned to its original orientation.

8:Polyobj_DoorSlide (po, speed, angle, dist, delay)

- po : polyobject to move
- speed: how quickly to move the polyobject
- angle: direction to move the polyobject in (this is a byte angle)
- dist : distance to move the polyobject
- delay: delay in tics before returning to original position

Moves a polyobject, waits, and then moves it back to its original location.

90:Polyobj_OR_RotateLeft (po, speed)

- po : polyobject to move
- speed: how quickly to move the polyobject
- angle: direction to move the polyobject in (this is a byte angle)

The OR in these specials stands for OverRide. Under normal circumstances, if a polyobject is doing something, you cannot make it do something else until it has finished whatever it is doing. This can be a problem with perpetual polyobjs (such as Polyobj_RotateLeft or Polyobj_RotateRight with a byte angle of 255). Using one of these four specials, you can force the polyobj to stop whatever it is doing and do something else.

91:Polyobj_OR_RotateRight (po, speed, angle)

- po : polyobject to move
- speed: how quickly to move the polyobject
- angle: direction to move the polyobject in (this is a byte angle)

The OR in these specials stands for OverRide. Under normal circumstances, if a polyobject is doing something, you cannot make it do something else until it has finished whatever it is doing. This can be a problem with perpetual polyobjects (such as Polyobj_RotateLeft or Right with a byte angle of 255). Using one of these four specials, you can force the polyobj to stop whatever it is doing and do something else.

92:Polyobj_OR_Move (po, speed, angle, distance)

- po : polyobject to move
- speed: how quickly to move the polyobject
- angle: direction to move the polyobject in (this is a byte angle)
- dist : distance to move the polyobject

The OR in these specials stands for OverRide. Under normal circumstances, if a polyobject is doing something, you cannot make it do something else until it has finished whatever it is doing. This can be a problem with perpetual polyobjects (such as Polyobj_RotateLeft or Polyobj_RotateRight with a byte angle of 255). Using one of these four specials, you can force the polyobject to stop whatever it is doing and do something else.

93:Polyobj_OR_MoveTimes8 (po, speed, angle, distance)

- po : polyobject to move
- speed: how quickly to move the polyobject
- angle: direction to move the polyobject in (this is a byte angle)
- dist : distance to move the polyobject

The OR in these specials stands for OverRide. Under normal circumstances, if a polyobject is doing something, you cannot make it do something else until it has finished whatever it is doing. This can be a problem with perpetual polyobjs (such as Polyobj_RotateLeft or Right with a byte angle of 255). Using one of these four specials, you can force the polyobj to stop whatever it is doing and do something else.

Platform Specials

60:Plat_PerpetualRaise (tag, speed, delay)

- tag: Tag of affected sector
- speed: Speed of move
- delay: Tics between rise and falls

Starts a perpetual raise platform with a lip of 8 units. If tag is 0, then the sector on the line’s back side is used.

61:Plat_Stop (tag)

- tag: Tag of affected sector

Stops a perpetual raise platform.

62:Plat_DownWaitUpStay (tag, speed, delay)

- tag: Tag of affected sector
- speed: Speed of move
- delay: Tics before going back up

Lowers a platform, waits, and then raises it back to its original position. This is the same as using Plat_DownWaitUpStayLip with a lip of 8. If tag is 0, then the sector on the line's back side is used.

63:Plat_DownByValue (tag, speed, delay, height)

- tag: Tag of affected sector
- speed: Speed of move
- delay: Tics before going back up
- height: Amount to lower by

Lowers a platform by (height * 8) units, waits, and then raises it back to its original position. If tag is 0, then the sector on the line's back side is used.

64:Plat_UpWaitDownStay (tag, speed, delay)

- tag: Tag of affected sector
- speed: Speed of move
- delay: Tics before going back down

Raises a platform, waits, and then lowers it back to its original position. If tag is 0, then the sector on the line's back side is used.

65:Plat_UpByValue (tag, speed, delay, height)

- tag: Tag of affected sector
- speed: Speed of move
- delay: Tics before going back down
- height: Amount to raise by

Raises a platform by (height * 8) units, waits, and then lowers it back to its original position. If tag is 0, then the sector on the line's back side is used.

172:Plat_UpNearestWaitDownStay (tag, speed, delay)

- tag: Tag of affected sector
- speed: Speed of move
- delay: Tics before going back dow

Raises the tagged sector to the nearest height, waits for delay tics, then lowers to its original height.

206:Plat_DownWaitUpStayLip (tag, speed, delay, lip, sound)

- tag: Tag of affected sector
- speed: Speed of move
- delay: Tics between rise and falls
- lip: Amount of floor to leave showing when platform is lowered

- sound: if 0 the platform sound is used, if 1 the moving floor sound is used

Lowers a platform, waits, and then raises it back to its original position. If tag is 0, then the sector on the line’s back side is used.

207:Plat_PerpetualRaiseLip (tag, speed, delay, lip)

- tag: Tag of affected sector
- speed: Speed of move
- delay: Tics between rise and falls
- lip: Amount of floor to leave showing when platform is lowered

Starts a perpetual raise platform. If tag is 0, then the sector on the line’s back side is used.

228:Plat_RaiseAndStayTx0 (tag, speed)

- tag: Tag of affected sector
- speed: Speed of move

Raises a platform, sets its floor texture to match the floor texture on the front side of the triggering line, and sets the plat’s sector special to zero. If tag is 0, then the sector on the line’s back side is used.

230:Plat_UpByValueStayTx (tag, speed, height)

- tag: Tag of affected sector
- speed: Speed of move
- height: Amount to raise by

Raises a platform by (height * 8) units, and sets its floor texture to match the floor texture on the front side of the triggering line. If tag is 0, then the sector on the line’s back side is used.

231:Plat_ToggleCeiling (tag)

- tag: Tag of affected sector

Switches a platform between lowered and touching the ceiling. If tag is 0, then the sector on the line’s back side is used.

Scroll Specials

100:Scroll_Texture_Left (speed)

- speed: Speed to scroll

Scrolls a texture to the left.

101:Scroll_Texture_Right (speed)

- speed: Speed to scroll

Scrolls a texture to the right.

102:Scroll_Texture_Up (speed)

- speed: Speed to scroll

Scrolls a texture up, with speed determined by the variable speed.

103:Scroll_Texture_Down (speed)

- speed: Speed to scroll

Scrolls a texture down, with speed determined by the variable speed.

221:Scroll_Texture_Both (lineid, left, right, down, up)

- lineid: Line ID of line to scroll (must be 0 for now)
- left: Speed to scroll left
- right: Speed to scroll right
- down: Speed to scroll down
- up: Speed to scroll up

Scrolls a texture both horizontally and vertically. If the lineid is 0, this special will affect the speed of the texture scrolling on the line it is placed on. The other four parameters determine how quickly the texture scrolls in each direction.

If you use a non-zero lineid, you can change the scrolling of the specified lines when this special is activated (either directly or in a script). A positive lineid changes the scrolling on the front of the line, and a negative lineid changes the scrolling on the back of the line.

222:Scroll_Texture_Model (lineid, scrollbits)

- lineid: the ID of the line to be affected
- scrollbits: Selects the type of scroller
  - bit 0(1): Displacement scroller
  - bit 1(2): Accelerative scroller

(If you know more about this special, please add to this description)

223:Scroll_Floor (tag, [scrollbits,] type, x-move, y-move)

- tag: Tag of affected sector
- scrollbits: How the scrolling speed is determined (see below)
- type: Set to 0 to scroll the floor texture. A setting of 1 pushes objects along the floor without scrolling the texture. Setting it to 2 scrolls both.
- x-move/y-move: how quickly and in what directions to scroll.
- scrollbits: Selects the type of scroller and how the speed and angle is determined. (Only when placed on a line)
  - bit 0(1): Displacement scroller
  - it 1(2): Accelerative scroller
  - bit 2(4): Use this lindef to get dx and dy

This special takes 5 arguments when placed on a line, but 4 arguments when used in a script.

Scroll_Floor can also only scroll the flat, only carry objects, or scroll and carry objects depending on what the type argument is set to. x-move and y-move are only used if scrollbits does not indicate to use the linedef to determine the scroll rate. When placed on a line, 128 is subtracted from these values to determine the actual direction and rate of scroll. (So 128 would be no scroll.) When used in a script, positive values move north/east, while negative values move south/west.

224:Scroll_Ceiling (tag, scrollbits, unused, x-move, y-move)

- tag: tag of affected sector
- scrollbits: how the scrolling speed is determined (see below)
- unused: this arg should always be set to 0.
- x-move/y-move: how quickly and in what directions to scroll.
- scrollbits: Selects the type of scroller and how the speed and angle is determined.
  - bit 0(1): Displacement scroller
  - bit 1(2): Accelerative scroller
  - bit 2(4): Use this lindef to get dx and dy

x-move and y-move are only used if scrollbits does not indicate to use the linedef to determine the scroll rate. 128 is subtracted from these values to determine the actual direction and rate of scroll. (So 128 would be no scroll.) Otherwise, the linedef's length and orientation determine the scroll rate and direction.

225:Scroll_Texture_Offsets

Scrolls a texture both horizontally and vertically. This special only affects the line it is being placed on. The amount of scrolling is determined from the texture offsets on the first side of this line. Only the first side can be scrolled with this special. This special is mainly provided for Boom compatibility.

Stair Specials

26:Stairs_BuildDown (tag, speed, height, delay, reset)

- tag: Tag of first sector in staircase
- speed: How quickly the steps lower
- height: Height of each step
- delay: Number of tics the staircase waits between steps
- reset: Tics until the stairs return to their original heights (0 if never)

Builds a staircase in sectors marked with the Stairs_Specials.

27:Stairs_BuildUp (tag, speed, height, delay, reset)

- tag: Tag of first sector in staircase
- speed: How quickly the steps rise
- height: Height of each step
- delay: Number of tics the staircase waits between steps
- reset: Tics until the stairs return to their original heights (0 if never)

Builds a staircase in sectors marked with the Stairs_Specials.

31:Stairs_BuildDownSync (tag, speed, height, reset)

- tag: Tag of first sector in staircase
- speed: How quickly the first step lowers
- height: Height of each step
- reset: Tics until the stairs return to their original heights (0 if never)

Builds a staircase in sectors marked with the Stairs_Specials. Each step moves at a different speed so that they all reach their destination heights at the same time.

217:Stairs_BuildUpDoom (tag, speed, height, delay, reset)

- tag: Tag of first sector in staircase
- speed: How quickly the steps rise
- height: Height of each step
- delay: Number of tics the staircase waits between steps
- reset: Tics until the stairs return to their original heights (0 if never)

Builds a staircase. The sectors that make up the staircase are determined using the normal Doom mechanism.

Thing Specials

Slope Specials

Several ways of creating slopes with thing specials are described below. Note that slopes can also be created by using the Plane Align line special.

1500:Floor slope (Z, arg1)

Used to describe a floor slope. It works together with the line of the sector where this thing is placed in, with Arg1 equal to the thing's TID number. The floor plane is defined with 3 points: the thing's position and the line ending vertices at the sector's floor height.

Hint: In WadAuthor if the line doesn't have a special type or the special type doesn't uses Arg1, Arg1 must be changed using "Edit raw data".

1501:Ceiling slope (Z, arg1)

Used to describe a ceiling slope. It works together with the line of the sector where this thing is placed in, with Arg1 equal to the thing's TID number. The ceiling plane is defined with 3 points: the thing's position and the line ending vertices at the sector's ceiling height.

Hint: In WadAuthor if the line doesn't have a special type or the special type doesn't uses Arg1, Arg1 must be changed using "Edit raw data".

1504:Vertex slope floor

Used to create a slope to a certain height at a vertex. It works only if all adjoining sectors are triangular, and height is specified by the Thing's z-height value.

1505:Vertex slope ceiling

Used to create a slope to a certain height at a vertex. It works only if all adjoining sectors are triangular, and height is specified by the Thing's z-height value.

9500:Point line slope floor

Another method used to describe a floor slope.

- The first argument to this thing is a line id.
- For each matching line, the sector on the same side of the line has its floor sloped so that it passes through this thing.
- Unlike thing 1500, this thing does not necessarily need to be in the sector(s) being sloped.

Hint: To give a line an id, set "special" to 121 and set Arg1 to the new line id.

9501:Point line slope ceiling

Another method used to describe a ceiling slope.

- The first argument to this thing is a line id.
- For each matching line, the sector on the same side of the line has its ceiling sloped so that it passes through this thing.
- Unlike thing 1501, this thing does not necessarily need to be in the sector(s) being sloped.

Hint: To give a line an id, set "special" to 121 and set Arg1 to the new line id.

9502:Direct set slope floor

Yet another method used to describe a floor slope.

This thing sets the plane equation for the sector its is in directly. The first argument determines how far it tilts from horizontal and is measured in degrees. 90 degrees is perfectly horizontal. Values close to 90 have less slope than values further from 90. The thing's angle is used to determine what direction the slope faces. The floor will be adjusted so that it passes through this thing.

9503:Direct set slope ceiling

Yet another method used to describe a ceiling slope.

This thing sets the plane equation for the sector its is in directly. The first argument determines how far it tilts from horizontal and is measured in degrees. 90 degrees is perfectly horizontal. Values close to 90 have less slope than values further from 90. The thing's angle is used to determine what direction the slope faces. The ceiling will be adjusted so that it passes through this thing.

9510:Copy slope plane floor

Copies the slope plane of the floor from one sector to another. The first argument to this is a sector tag. The plane equation from the first sector with a matching tag will be copied to the sector that this thing is in. This thing overrides any slopes that might have been created in their sectors already.

9511:Copy slope plane ceiling

Copies the slope plane of the ceiling from one sector to another. The first argument to this is a sector tag. The plane equation from the first sector with a matching tag will be copied to the sector that this thing is in. This thing overrides any slopes that might have been created in their sectors already.

Light Specials

1502:Static light

- z: Z position of the light source
- arg1: octradius (i.e. radius is Arg1 * 8)

Adds a static light source at a given position with the given radius. If arg1 is 0, a default value is used.

1503:Static colored light

- z: Z position of the light source
- arg1: octradius (i.e. radius is Arg1 * 8)
- arg2: R
- arg: G
- arg4: B

Adds a static colored light source at a given position with a given radius and color. R is red, G – green and B – blue component of the light color.

Teleport Specials

39:Teleport_ZombieChanger (tid, tag)

- tid: Thing ID of the destination spot
- tag: Tag of destination sector

Teleports the activating thing to a new location, but without fog or a delay. After teleporting the teleported object switches to its pain state.

If tag is 0, it will use a random teleport destination out of those with the matching tid. This can be restricted to certain sectors if tag!=0. If Tid is 0 and Tag!=0 it will use the first teleport destination found in the first sector with the matching tag.

70:Teleport (tid, tag, nosourcefog)

- tid: Thing ID of the destination spot
- tag: Tag of the destination sector

- nosourcefog: If !=0 only spawns a teleport fog object at the destination but not the source of the teleportation

Teleports the activating thing to a new location. If tag is 0, it will use a random teleport destination out of those with the matching tid. This can be restricted to certain sectors if tag!=0. If Tid is 0 and Tag!=0 it will use the first teleport destination found in the first sector with the matching tag.

71:Teleport_NoFog (tid, useangle, tag)

- tid: Thing ID of the destination spot
- useangle: if !=0 the orientation of the teleported object is not preserved.
- tag: Tag of destination sector

Teleports the activating thing to a new location, but without fog or a delay, and preserves the thing's facing angle relative to the triggering line and the destination spot. This can be overridden by passing a non-zero value as the second parameter. In this case the teleported object will face in the same direction as the teleport destination spot.

If tag is 0, it will use a random teleport destination out of those with the matching tid. This can be restricted to certain sectors if tag!=0. If Tid is 0 and Tag!=0 it will use the first teleport destination found in the first sector with the matching tag.

74:Teleport_NewMap (map, pos, face)

- map: Levelnum of the map to teleport to
- pos: Corresponds to destination player start spot arg0
- face: If TRUE (> 0) then the player will retain the direction they were facing from the last map

Teleports the player to a new map and to the player start spot whose arg0 matches pos. Example:

```
Teleport_NewMap (2, 0, 1);
```

In Doom2 wads in Hexen format (ZDoom) the above line means teleport the player to map02 (underhalls) and the player 1 start (0) keeping the direction they were facing on the previous map.

75:Teleport_EndGame

Ends the game.

76:TeleportOther(tid, destination tid, fog?)

- tid: TID of the thing to teleport.
- destination tid: TID of the map spot to teleport to.
- fog?: Teleport with or without fog. (0 = no fog, 1 = fog)

Teleports a thing. The special's main usage is to teleport something else besides a player.

77:TeleportGroup(group tid, source tid, destination tid, move source?, fog?)

- group tid: The TID of the thing(s) to teleport. If 0, teleports the activator.
- source tid: The spot relative to which to teleport.
- destination tid: The destination spot relative to which to spawn.
- move source?: Teleport the source TID as well? (0 = no, 1 = yes)
- fog?: Spawn with or without fog. (0 = no fog, 1 = fog)

Teleport a group of actors. Source_tid is used as a reference point so that they end up in the same position relative to dest_tid.

78:TeleportInSector (tag, source tid, destination tid, fog?, group tid)

- tag: Tag of the sector from which will be teleported.
- source tid: The spot relative to which to teleport.
- destination tid: The destination spot relative to which to spawn.
- fog?: Spawn with or without fog. (0 = no fog, 1 = fog)
- group tid: The TID of the thing(s) to teleport. If 0, teleports all actors in the sector

Teleport a group of actors in a sector. Source_tid is used as a reference point so that they end up in the same position relative to dest_tid. Group_tid can be used to not teleport all actors in the sector.

Miscellaneous Specials

72:ThrustThing (angle, force)

- angle: Byte angle to thrust the thing
- force: How far to thrust the thing

Thrusts the thing that activated the special.

73:DamageThing (amount)

- amount: Amount of damage to do

Hurts the thing that activated the special. If amount is 0, then the thing is guaranteed to be killed.

130:Thing_Activate (tid)

- tid: Thing ID of the thing to activate

Activates a thing so it can be used. In the case of a monster, removes the dormant flag from a monster so that it will be able to take damage and attack the player.

131:Thing_Deactivate (tid)

- tid: Thing ID of the thing to deactivate

Sets the dormant flag of a monster so that it won't be able to take damage or attack the player.

132:Thing_Remove (tid)

- tid: Thing ID of the thing to remove

Removes the specified thing from the map.

133:Thing_Destroy (tid, extreme)

- tid: Thing ID of the thing to destroy

Kills the specified thing. If you have a TID of '0' then it will kill all the monsters on the map, similar to the 'massacre' cheat. If extreme is 1 the thing enters its extreme (gib) death sequence if it exists.

Note that this function does not properly handle killing players with armor or more than 100% health if extreme is 0.

134:Thing_Projectile (tid, type, angle, speed, vspeed)

- tid: Thing ID of the map spot to spawn the projectile at
- type: Type of projectile to spawn
- angle: Byte angle of the projectile

- speed: Speed of the projectile in the x-y plane
- vspeed: Vertical speed of the projectile (up is positive)

Spawns a projectile.

Usage:

TID is a number that makes a thing unique, the default is zero, and therefore a TID is usually set to something else, like 1. In DoomBuilder you have to right click on a thing, select the Effects tab and change the Thing Tag to something besides 0.

The **type** is actually from the list of Spawn_numbers. You can either use the number or the defined name (T_SOMETHING) for clarity.

Angle is a number between 0 and 255. **Speed** is a number between 0 and 255. **Vspeed** is a number between -255 and 255.

135:Thing_Spawn (tid, type, angle, new tid)

- tid: Thing ID of the map spot to spawn the thing at type: Type of thing to spawn
- angle: Byte angle for the thing to face
- new tid: TID to give spawned thing

Spawns a thing.

Usage:

TID is a number that makes a thing unique, the default is zero, and therefore a TID is usually set to something else, like 1. In DoomBuilder you have to right click on a thing, select the Effects tab and change the Thing Tag to something besides 0.

The **type** is actually from the list of Spawn_numbers. You can either use the number or the defined name (T_SOMETHING) for clarity.

Angle is a number between 0 and 255.

New TID is just a way to make the spawned thing unique if you need to refer to it later. It is just a number.

136:Thing_ProjectileGravity (tid, type, angle, speed, vspeed)

- tid: Thing ID of the map spot to spawn the projectile at
- type: Type of projectile to spawn
- angle: Byte angle of the projectile
- speed: Speed of the projectile in the x-y plane
- vspeed: Vertical speed of the projectile (up is positive)

Spawns a projectile and subjects it to gravity.

TID is a number that makes a thing unique, the default is zero, and therefore a TID is usually set to something else, like 1. In DoomBuilder you have to right click on a thing, select the Effects tab and change the Thing Tag to something besides 0.

The **type** is actually from the list of Spawn_numbers. You can either use the number or the defined name (T_SOMETHING) for clarity.

Angle is a number between 0 and 255. **Speed** is a number between 0 and 255. **Vspeed** is a number between -255 and 255.

137:Thing_SpawnNoFog (tid, type, angle, new tid)

- tid: Thing ID of the map spot to spawn the thing at
- type: Type of thing to spawn
- angle: Byte angle for the thing to face
- new tid: TID to give spawned thing

Spawns a thing without the teleporter fog.

Usage:

TID is a number that makes a thing unique, the default is zero, and therefore a TID is usually set to something else, like 1. In DoomBuilder you have to right click on a thing, select the Effects tab and change the Thing Tag to something besides 0.

The **type** is actually from the list of Spawn_numbers. You can either use the number or the defined name (T_SOMETHING) for clarity.

Angle is a number between 0 and 255.

New TID is just a way to make the spawned thing unique if you need to refer to it later. It is just a number.

Other Specials

33:ForceField (No parameters required)

Turns the line into a forcefield that will hurt (8 damage on SKILL_VERY_EASY, 16 on all other) and push the player when activating. It does *not* block the player. Note that you will have to set it “repeatable” to make the force field persistent and you have to set the proper activation type.

34:ClearForceField (tag)

Clears all force fields from lines connected to the tagged sector. It will also clear the force field’s middle texture.

95:FloorAndCeiling_LowerByValue (tag, speed, height)

- tag: Tag of affected sector
- speed: Speed of the move
- height: Amount to move

Moves both the floor and ceiling of the affected sectors down by height units.

96:FloorAndCeiling_RaiseByValue (tag, speed, height)

- tag: Tag of affected sector
- speed: Speed of the move
- height: Amount to move

Moves both the floor and ceiling of the affected sectors up by height units.

120:Radius_Quake (intensity, duration, damrad, tremrad, tid)

- intensity: Strength of earthquake [1..9]
- duration: Duration in tics
- damrad: Radius of damage in 64×64 cells
- tremrad: Radius of tremor in 64×64 cells
- tid: Thing ID of map thing(s) for quake foci

Creates an earthquake at the specified foci (map spots).

121:Line_SetIdentification (lineid)

- lineid: Identification number for this line

Used to identify this line for certain specials and ACS commands.

129:UsePuzzleItem (item, script)

- item: Puzzle artifact number
- script: Script to execute

Executes a specified script if player holds specified puzzle artifact.

Puzzle artifacts (class names):

- 0 Skull
- 1 GemBig
- 2 GemRed
- 3 GemGreen1
- 4 GemGreen2
- 5 GemBlue1
- 6 GemBlue2
- 7 Book1
- 8 Book2
- 9 FlameMask
- 10 FWeapon
- 11 CWeapon
- 12 MWeapon
- 13 Gear1
- 14 Gear2
- 15 Gear3
- 16 Gear4

This special can also be put on any thing in the map. If that is the case and the player uses the item near this thing the special is also executed.

140:Sector_ChangeSound (tag, newsequence)

- tag: Tag of affected sector
- newsequence: Number of new sound sequence

Changes the sound sequence attached to the sector. This is the same as the one specified by placing one of the sound sequence things in a map editor but with this the sequence can be changed at run time.

160:Sector_Set3DFloor (tag)

Adds an extra floor in all sectors with tag equal to Arg1. The extra floor is described with the line's front sector. Its floor describes the floor of the top region, its ceiling the ceiling of bottom region; light level and contents: light level and contents of the bottom region.

Warning: WadAuthor checks treat floors lower than ceilings as error, so for maps with 3D floors you should never use "Fix all".

161:Sector_SetContents (type, translucency, flags)

Defines contents type of the front sector.

Content types:

- 0 – empty
- 1 – water
- 2 – lava
- 3 – nukage
- 4 – slime
- 5 – hellslime
- 6 – blood
- 7 – sludge
- 8 – hazard
- 9 – BOOM water

Translucency is given in percents, 0 – solid, 100 – invisible.

Flags applies to both floor and ceiling and are a combination of the following values:

1 – don't block movement
2 – don't block sight
4 – don't block shooting
8 – draws the 3D floor using additive translucency

Translucency and flags have sense only if the sector is used for 3D floors. Usualy for solid 3D floors you will have flags 0, for water surfaces 7.

173:NoiseAlert (target_tid, emitter_tid)

This special alerts monsters of a target's presence. You can specify both the object which is doing this alert and the target which the monsters should attack. If both parameters are 0, the special's activator is both emitter and target.

Which monsters are woken up is determined in a similar way to the player shooting: lines with the "block sounds" flag and sectors with 0 height will not allow the imaginary sound to travel past them. If the monster's "deaf" flag is set, it will not wake up immediately, but as soon as the player comes in line of sight of the actor, no matter what direction the monster is actually facing.

181:Plane_Align(floor, ceiling)

- floor: Give this a value of 1 and the sector on the front side will be sloped; giving it a value of 2 slopes the sector on the back side. Note: A value of 0 means no sloping is applied.
- ceiling: Give this a value of 1 and the sector on the front side will be sloped; giving it a value of 2 slopes the sector on the back side. Note: A value of 0 means no sloping is applied.

Aligns the plane of the floor and/or ceiling depending on the values "floor" and "ceiling".

208:TranslucentLine (lineid, amount, additive?)

- lineid: Line ID of lines to make translucent (0 for this line)
- amount: How translucent the line should be [0..255].
- additive?: Whether this translucent line should use additive translucency or not (0 = normal, 1 = additive)

Sets the amount of translucency for all matching lines (including itself). If lineid is 0, it only sets the translucency of the line it is on. Like Line_SetIdentification, this special sets the linedef's id. Amount controls how opaque the line is. 0 is nearly invisible, 255 is opaque. Intermediate values are somewhere in between.

You can also use this special in an ACS script to change the translucency for lines whose ids match lineid.

210:Transfer_FloorLight (tag)

- tag: Tag of affected sector

The amount of lighting on the tagged sector's floor will be the same as the lighting in the sector that this line faces.

211:Transfer_CeilingLight (tag)

- tag: Tag of affected sector

The amount of lighting on the tagged sector's ceiling will be the same as the lighting in the sector that this line faces.

213:Sector_SetFade (tag, r, g, b)

- tag: Tag of affected sector
- r: Amount of red to fade to
- g: Amount of green to fade to
- b: Amount of blue to fade to

Sets the color that light in the tagged sectors fade to as they get further away from the viewer. This will give the sector a 'fog' effect and can greatly enhance the mood of a level. By default, this is whatever the level's fadeto is specified as being in a MAPINFO lump, or black, if it doesn't specify a fadeto. The note about changing a sector's color during the middle of gameplay also applies here. You can use the testfade console command to test a fade from within the game. Keep in mind that Sector_SetFade is affected by a sector's light, so a faded sector with a light value of 255 will have virtually no visible fade color, and a faded sector with a light value of 0 will be nearly a solid color.

214:Sector_SetDamage (tag, amount, mod)

- tag: Tag of affected sector
- amount: Amount of damage to apply
- mod: Means-of-death identifier
  - 0 = MOD_UNKNOWN
  - 12 = MOD_WATER
  - 13 = MOD_SLIME
  - 14 = MOD_LAVA
  - 15 = MOD_CRUSH
  - 16 = MOD_TELEFRAG
  - 17 = MOD_FALLING
  - 18 = MOD_SUICIDE
  - 19 = MOD_BARREL
  - 20 = MOD_EXIT
  - 21 = MOD_SPLASH
  - 22 = MOD_HIT

Sets the amount of damage done to a player in a sector. This is in addition to whatever is specified by the sector's special. Damage amount below 20 will only hurt the player if he doesn't have an environment suit. Damages between 20-49 will occasionally hurt the player even with an environment suit on. Damages of 50 and above will always hurt the player unless he is in god mode.

227:PointPush_SetForce (tag, tid, amount, useline)

- tag: Tag of sector containing the point pusher/puller
- tid: Thing ID of the point pusher/puller
- amount: Strength of the point pusher/puller
- useline: Set to 1 to use the line's length to determine strength

Only valid during initialization. Sets the amount of force for a point pusher or puller. If tag is non-zero it looks for point pusher/pullers in the tagged sectors. Otherwise, it looks for them by their tids. If useline is 1, then the amount of force is determined by the length of the linedef. Otherwise, the amount parameter is used.

243:Exit_Normal (pos)

- pos: Corresponds to destination player start spot arg0

Teleports the player to the next map defined for this map in MAPINFO and to the player start spot whose arg0 matches pos.

245:Elevator_RaiseToNearest (tag, speed)

- tag: Tag of affected sector
- speed: How fast the elevator moves

Raises the floor and ceiling of the tagged sector so that its floor aligns with the floor of the next higher sector while keeping the same amount of distance between the floor and ceiling.

246:Elevator_MoveToFloor (tag, speed)

- tag: Tag of affected sector
- speed: How fast the elevator moves

Moves the floor and ceiling of the tagged sector so that its floor aligns with the floor of the sector the activating linedef is facing, while keeping the same amount of distance between the floor and ceiling.

247:Elevator_LowerToNearest (tag, speed)

- tag: Tag of affected sector
- speed: How fast the elevator moves

Lowers the floor and ceiling of the tagged sector so that its floor aligns with the floor of the next lower sector while keeping the same amount of distance between the floor and ceiling.

251:FloorAndCeiling_LowerRaise (tag, fspeed, cspeed)

- tag: Tag of affected sector
- fspeed: Speed to move the floor
- cspeed: Speed to move the ceiling

Lowers the sector’s floor to the lowest surrounding floor and raises its ceiling to the highest surrounding ceiling.

BOOM Specials

200:Generic_Floor (tag, speed, height, target, flags)

- tag: Tag of affected sector
- speed: How quickly the floor moves
- height: Height of the move (for types that use it)
- target: How to determine the floor’s destination height
- flags: Flags for the move

This special encapsulates BOOM’s generalized floors.

Flags are:

Value	Meaning
0 (0x00)	No change
1 (0x01)	Zero sector’s special

2 (0x02)	Change sector’s floor picture
3 (0x03)	Change sector’s special
4 (0x04)	Use numeric model if set, trigger model if not 8 (0x08) Raise floor if set, lower it if not
16 (0x10)	Cause crushing damage if set

Target can be one of the following:

Value	Target
0	Height units above/below current height
1	Highest surrounding floor
2	Lowest surrounding floor
3	Nearest surrounding floor
4	Lowest surrounding ceiling
5	This sector’s ceiling
6	Height units determined by shortest lower texture on sector

If tag is 0, then the sector on the line’s back side is used.

201:Generic_Ceiling (tag, speed, height, target, flag)

- tag: Tag of affected sector
- speed: How quickly the ceiling moves
- height: Height of the move (for types that use it)
- target: How to determine the ceiling’s destination height
- flags: Flags for the move

This special encapsulates BOOM’s generalized ceilings.

Flags are:

Value	Meaning
0 (0x00)	No change
1 (0x01)	Zero sector’s special
2 (0x02)	Change sector’s ceiling picture
3 (0x03)	Change sector’s special
4 (0x04)	Use numeric model if set, trigger model if not 8 (0x08) Raise ceiling if set, lower it if not
16 (0x10)	Cause crushing damage if set

Target can be one of the following:

Value	Target
0	Height units above/below current height

1	Highest surrounding ceiling
2	Lowest surrounding ceiling
3	Nearest surrounding ceiling
4	Highest surrounding floor
5	This sector's floor
6	Height units determined by shortest upper texture on sector

If tag is 0, then the sector on the line's back side is used.

202:Generic_Door (tag, speed, kind, delay, lock)

- tag: Tag of affected sector
- speed: How quickly the door moves
- kind: What type of door this is. Kind can be one of four values:
  - 0: Raise door and close it after delay octics
  - 1: Open door and leave it open
  - 2: Close door and open it after delay octics
  - 3: Close door and leave it closed
- delay: Octics until door is automatically closed/opened lock: Required key, if any (see [[Key types]])

This special encapsulates BOOM's generalized doors.

If 128 is added to the kind parameter the tag is used as a light tag, not a door tag. In this case this special only opens the door on the line's back side.
Instead a gradual lighting effect is done in the tagged sectors. The light is gradually changed between the darkest neighboring sector when the door is fully closed and the brightest neighboring sector when the door is fully open.

If tag is 0, then the sector on the line's back side is used.

203:Generic_Lift (tag, speed, delay, type, height)

- tag: Tag of affected sector
- speed: Speed of move
- delay: Tics before reversing direction
- type: Specifies what type of platform this is. Type can be one of the following values:
  - 0: Plat_UpByValue
  - 1: Plat_DownWaitUpStay
  - 2: Plat_DownToNearestFloor
  - 3: Plat_DownToLowestCeiling
  - 4: Plat_PerpetualRaise
- height: Distance to move if target is 0.

This special encapsulates BOOM's generalized lifts.

204:Generic_Stairs (tag, speed, height, flags, reset

- tag: Tag of first sector in staircase
- speed: How quickly the steps move
- height: Height of each step
- flags: controls direction and ignorance of floor textures
- reset: Tics until the stairs return to their original heights (0 if never)

This special encapsulates BOOM's generalized stairs. The sectors of the staircase are determined as they are in BOOM. If bit 0 of flags is set (0x01), the stairs are built up, otherwise they build down. If bit 1 is set (0x02), then the determination of stair sectors ignores different floor textures. If a line activates this special, and the line is marked as repeatable, then the direction the stairs build will alternate between up and down each time the special is activated.

205:Generic_Crusher (tag, dspeed, uspeed, silent, crush)

- tag: Tag of affected sector
- dspeed: How quickly the ceiling lowers
- uspeed: How quickly the ceiling raises
- silent: Set this to 1 to prevent the crusher from making noise
- crush: Amount of damage to apply

This special encapsulates BOOM's generalized crushers. It is the same as Ceiling_CrushAndRaiseA except that it takes an argument to control the "noisiness" of the crusher.

209:Transfer_Heights (tag, flags)

- tag: Tag of affected sector
- flags: Defines the properties of the effect

This is BOOM's 242 linedef. It is used to divide a sector into upper, lower, and middle regions. The drawn heights of the tagged sector's floor and ceiling come from the heights of the sector on the line's front side. The line's texture names can be used to specify a special color map or palette blend to be used in those regions. Palette blends are of the form "AARRGGBB". You can use the testblend command to find a good blend from inside the game.

The **flags** parameter determines how the fake floor and ceiling are rendered and can have values such as:

- 0 = There will be situations when the sector's real ceiling and floor heights will be used instead of the fake heights (this is the way BOOM does it).
- 1 = The fake ceiling and floor heights will always be used.
- 2 = Only the fake floor will be drawn.
- 4 = The fake floor/ceiling is only visible if it is inside of the sector the player moves around in. The texture on the control sector's floor is also used for the top and bottom of the fake floor. The same applies to the fake ceiling.
- 8 = Make the target sector swimmable under the fake floor (no need for the WaterZone thing).
- 16 = Do not draw the fake floor (or ceiling); however they can still be used in conjunction with the sector action things that correspond to them (useful for making the player fall down a deep hole or similar).
- 32 = Do not transfer the control sector's lighting to the affected sector(s).

These values can also be combined by adding them. For example a value of 6 would combine values 2 and 4, and a value of 29 would combine values 16, 8, 4 and 1.

The following description is an excerpt from boomref.txt:

This allows the tagged sector to have two levels – an actual floor and ceiling, and another floor or ceiling where more flats are rendered. Things will stand on the actual floor or hang from the actual ceiling, while this function provides another rendered floor and ceiling at the heights of the sector on the first sidedef of the linedef. Typical use is "deep water" that can be over the player's head. [Note: See the waterzone thing (9045)]

```
---------------------------------- < real sector's ceiling height
|              real ceiling              | < control sector's ceiling texture
|                                        |
|                                        | < control sector's lightlevel
|                   A                    |
|                                        | < upper texture as colormap
|                                        |
|                                        | < control sector's floor texture
---------------------------------- < control sector's ceiling height
```

```
|              fake ceiling              | < real sector's ceiling texture
|                                        |
|                                        | < real sector's lightlevel
|                  B                     |
|                                        | < normal texture as colormap
|                                        |
|               fake floor               | < real sector's floor texture
---------------------------------- < control sector's floor height
|                                        | < control sector's ceiling texture
|                                        |
|                                        | < control sector's lightlevel
|                  C                     |
|                                        | < lower texture as colormap
|                                        |
|               real floor               | < control sector's floor texture
---------------------------------- < real sector's floor height
```

Boom sectors controlled by a 242 linedef are partitioned into 3 spaces. The viewer's xyz coordinates uniquely determine which space they are in.

If they are in space B (normal space), then the floor and ceiling textures and lightlevel from the real sector are used, and the colormap from the 242 linedef's first sidedef's normal texture is used (COLORMAP is used if it's invalid or missing). The floor and ceiling are rendered at the control sector's heights.

If they are in space C ("underwater"), then the floor and ceiling textures and lightlevel from the control sector are used, and the lower texture in the 242 linedef's first sidedef is used as the colormap.

If they are in space A ("head over ceiling"), then the floor and ceiling textures and lightlevel from the control sector are used, and the upper texture in the 242 linedef's first sidedef is used as the colormap.

If only two of these adjacent partitions in z-space are used, such as underwater and normal space, one has complete control over floor textures, ceiling textures, light level, and colormaps, in each of the two partitions. The control sector determines the textures and lighting in the more "unusual" case (e.g. underwater).

It's also possible for the fake floor to extend below the real floor, in which case an invisible platform / stair effect is created. In that case, the picture looks like this (barring any ceiling effects too):

```
---------------------------------- < real sector's ceiling texture
|       real ceiling = fake ceiling      |
|                                        |
|                                        |
|                  B                     | < real sector's lightlevel
|                                        | < normal texture's colormap
|                                        |
|               real floor               |
---------------------------------- < invisible, no texture drawn
|                                        |
|                                        |
|                                        | < real sector's lightlevel
|                  C                     | < normal texture's colormap
|                                        |
|                                        |
|               fake floor               | < real sector's floor texture
---------------------------------- < fake sector's floor height
```

In this case, since the viewer is always at or above the fake floor, no colormap/lighting/texture changes occur – the fake floor just gets drawn at the control sector's height, but at the real sector's lighting and texture, while objects stand on the higher height of the real floor.

It's the viewer's position relative to the fake floor and/or fake ceiling which determines whether the control sector's lighting and textures should be used, and which colormap should be used. If the viewer is always between the fake floor and fake ceiling, then no colormap, lighting, or texture changes occur, and the view just sees the real sector's textures and light level drawn at possibly different heights.

If the viewer is below the fake floor height set by the control sector, or is above the fake ceiling height set by the control sector, then the corresponding colormap is used (lower or upper texture name), and the textures and lighting are taken from the control sector rather than the real sector. They are still stacked vertically in standard order – the control sector's ceiling is always drawn above the viewer, and the control sector's floor is always drawn below the viewer.

The kaleidescope effect only occurs when F_SKY1 is used as the control sector's floor or ceiling. If F_SKY1 is used as the control sector's ceiling texture, then under water, only the control sector's floor appears, but it "envelops" the viewer. Similarly, if F_SKY1 is used as the control sector's floor texture, then when the player's head is over a fake ceiling, the control sector's ceiling is used throughout.

F_SKY1 causes HOM when used as a fake ceiling between the viewer and normal space. Since there is no other good use for it, this kaleidescope is an option turned on by F_SKY1. Note that this does not preclude the use of sky REAL ceilings over deep water – this is the control sector's ceiling, the one displayed when the viewer is underwater, not the real one.

A colormap has the same size and format as Doom's COLORMAP. Extra colormaps may be defined in Boom by adding them between the C_START and C_END markers in wads. Colormaps between C_START and C_END are automatically merged by Boom with any previously defined colormaps.

Ceiling bleeding may occur if required upper textures are not used.

Example wad: http://www.doomworld.com/idgames/index.php?id=12456

Sector Specials

- 1:LightPhased
- 2:LightSequenceStart
- 3:LightSequence
- 4:LightSequenceAlt
- 26:StairsSpecial1
- 27:StairsSpecial2
- 40:WindEastSlow
- 41:WindEastMedium
- 42:WindEastFast
- 43:WindNorthSlow
- 44:WindNorthMedium
- 45:WindNorthFast
- 46:WindSouthSlow
- 47:WindSouthMedium
- 48:WindSouthFast
- 49:WindWestSlow
- 50:WindWestMedium
- 51:WindWestFast
- 65:LightFlicker
- 66:LightStrobeFast
- 67:LightStrobeSlow
- 68:LightStrobeFastDamage
- 69:DamageHellslime
- 70:DamageSludge
- 71:DamageNukage
- 72:LightGlow
- 74:DoorCloseIn30
- 75:DamageSuperHellslimeExit
- 76:LightSyncStrobeSlow
- 77:LightSyncStrobeFast

- 78:DoorRaiseIn5Minutes
- 79:FrictionLow
- 80:DamageSuperHellslime
- 81:LightFireFlicker
- 82:DamageLavaWimpy
- 83:DamageLavaHefty
- 84:ScrollEastLavaDamage
- 105:DamageHazard
- 115:DamageInstantDeath
- 116:DamageSuperHazard
- 118:ScrollCurrent
- 197:LightningOutdoor
- 198:LightningIndoor1
- 199:Lightningindoor2
- 200:Sky2
- 201:ScrollNorthSlow
- 202:ScrollNorthMedium
- 203:ScrollNorthFast
- 204:ScrollEastSlow
- 205:ScrollEastMedium
- 206:ScrollEastFast
- 207:ScrollSouthSlow
- 208:ScrollSouthMedium
- 209:ScrollSouthFast
- 210:ScrollWestSlow
- 211:ScrollWestMedium
- 212:ScrollWestFast
- 213:ScrollNorthWestSlow
- 214:ScrollNorthWestMedium
- 215:ScrollNorthWestFast
- 216:ScrollNorthEastSlow
- 217:ScrollNorthEastMedium
- 218:ScrollNorthEastFast
- 219:ScrollSouthEastSlow
- 220:ScrollSouthEastMedium
- 221:ScrollSouthEastFast
- 222:ScrollSouthWestSlow
- 223:ScrollSouthWestMedium
- 224:ScrollSouthWestFast
- 225:ScrollEast5
- 226:ScrollEast10
- 227:ScrollEast25
- 228:ScrollEast30
- 229:ScrollEast35
- 230:ScrollNorth5
- 231:ScrollNorth10
- 232:ScrollNorth25
- 233:ScrollNorth30
- 234:ScrollNorth35
- 235:ScrollSouth5
- 236:ScrollSouth10
- 237:ScrollSouth25
- 238:ScrollSouth30
- 239:ScrollSouth35
- 240:ScrollWest5
- 241:ScrollWest10
- 242:ScrollWest25

- 243:ScrollWest30
- 244:ScrollWest35
- 0x0300:DAMAGE_MASK
- 0x0400:SECRET_MASK
- 0x0800:FRICTION_MASK
- 0x1000:PUSH_MASK