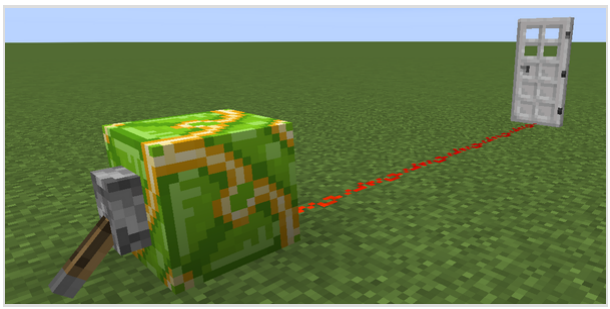
# Redstone circuits



A **redstone circuit** is a build that is primarily focused on dealing with logic and redstone signals. Circuits can be used to transmit redstone signals, perform operations with logic gates, store information, shorten or lengthen redstone signals, make clocks, and more.



A simple example of the effect of redstone power.

A useful distinction can be made between a **circuit** performing operations on signals (generating, modifying, combining, etc.), and a **mechanism** using the environment (moving blocks, opening doors, changing the light level, producing sound, etc). Making this distinction allows us to talk about the various circuits separately, and let all players choose whichever circuits are useful for their purposes. The machines controlled by redstone circuits can range from simple devices such as automatic doors and light switches to complex devices such as elevators, automatic farms, or even in-game computers. However, *this* article provides only an overview of redstone *circuits* as above. These can be used to control simple mechanisms, or combined as parts of a larger build. Each circuit type on this page has links to its own page, which provides greater detail about them and give schematics for multiple variations of each.

Before working with any but the most basic redstone circuits, an understanding of some basic concepts is required: "power", "signal strength", "redstone ticks", "block updates", etc. Some relevant articles are listed below:

- The Redstone mechanics article provides more information on these concepts.
- The Redstone components article adds a list and description of all blocks that interact with redstone power in general.
- The Mechanisms tutorial complements this article with an assortment of mechanism designs using circuits described here.
- The Redstone tips tutorial gives general advice about building redstone mechanisms

## Contents

# Describing circuits

Most circuits are described using Schematic diagrams; some of these require multiple images to show one or two layers per image. See the Help:Schematic page for details on how various blocks and components are represented.

## Size

The wiki describes circuit size (the volume of the rectangular solid it occupies) with the notation of *depth* × *width* × *height*, including support/floor blocks, but not including inputs/outputs.

Another method used for describing circuit size in the *Minecraft* community is to ignore non-redstone blocks simply used for support (for example, blocks under redstone dust or repeaters). However, this method is unable to distinguish between flat and 1-high circuits, as well as some other circuit differences.

Sometimes it is convenient to compare circuits simply by the area of their footprint (e.g., 3×4 for a circuit three-block wide by four blocks long), or by a single dimension important in a particular context (e.g., length in a sequence of sub-circuits, height in a confined space, etc.).

## Features

Several features may be considered desirable design goals:

**1-high**
: A circuit is 1-high (aka "1-tall") if its vertical dimension is one block high (meaning it can't have any redstone components that require support blocks under them, like redstone dust, repeaters, etc). Also see flat.

**1-wide**
: A circuit is 1-wide if at least 1 of its horizontal dimensions is exactly 1 block wide.

**Flat**
: A circuit is flat if it generally can be laid out on the ground with no components above another (support blocks under components are okay). Flat structures are usually easier for beginners to understand and build, and fit nicely under floors or on top of roofs. Also see 1-high.

**Flush**

A circuit is flush if it doesn't extend beyond a flat wall, floor, or ceiling and can still provide utility to the other side, though redstone mechanisms can be visible in the wall. Flush is a design goal for piston-extenders, piston doors, etc. Also see hipster and seamless.

**Hipster**
A circuit is hipster if it is initially hidden behind a flat wall, floor, or ceiling and can still provide utility to the other side. See also flush and seamless.

**Instant**
A circuit is instant if its output responds immediately to the input (no delay).

**Seamless**
A circuit is seamless if no redstone components are visible both before and after it completes its task (but it's okay if some are visible during operation). Seamless is a desirable design goal for piston-extenders, piston doors, etc. See also flush and hipster.

**Silent**
A circuit is silent if it doesn't make noise (such as from piston movement, dispenser/dropper activating when empty, etc.). Silent structures are desirable for traps or peaceful homes.

**Stackable**
A circuit is stackable if it can be placed "directly" on top of other copies of itself, and they all can be controlled as a single unit. Also see tileable.

**Expandable**
A circuit is Expandable if it can be placed "directly" next to other copies of itself, and they all can be controlled as a single unit. Also see tileable.

**Tileable**
A circuit is tileable if it can be placed "directly" next to or on top of other copies of itself, and each copy can still be controlled independently. Also see stackable.
Circuits might be described as "2-wide tileable" (tileable every two spaces in one dimension), or "2×4 tileable" (tileable in two directions), etc. Some structures might be described as "alternating tileable", meaning they can be placed next to each other if every other one is flipped or a slightly different design.

Other design goals may include reducing the delay a sub-circuit adds to a larger circuit, reducing the use of resource-expensive components (redstone, nether quartz, etc.), and re-arranging or redesigning a circuit to make it as small as possible.

Some components are not available before a player has access to the Nether, which limits the designs available. In particular, redstone comparators, observers and daylight detectors require nether quartz, which is available only from the Nether. Additionally, redstone lamps require glowstone, which is occasionally available from trading or witches, but is much more plentiful in the Nether.

# Circuit types

Although the number of ways to construct circuits is endless, certain patterns of construction occur repeatedly. The following sections attempt to categorize the circuits that have proven useful to the *Minecraft* community, while the main articles describe the specific circuits that fall into those categories.

Some of these circuits might be used by themselves for simple control of mechanisms, but frequently the player needs to combine them into more complex circuits to meet the needs of a

mechanism.

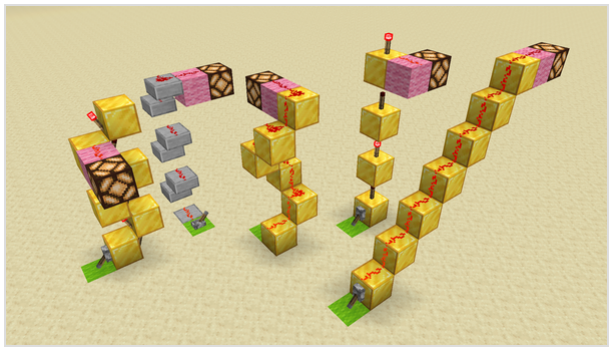
## Transmission circuit

*Main article: Transmission circuit*

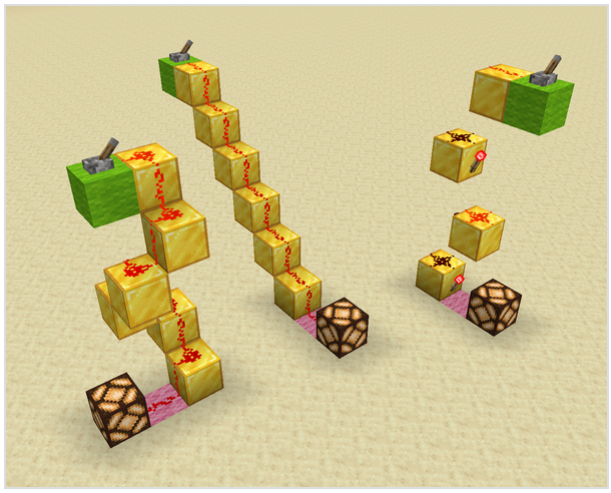
Some aspects of signal transmission can be helpful to understand: transmission types, vertical transmission, repeaters, and diodes.

### Vertical transmission

Although horizontal signal transmission is straightforward, vertical transmission involves options and trade-offs.



Transmitting signals upward



Transmitting signals downward

#### Redstone staircases

The simplest way but not the best way to transmit signals vertically is by placing redstone dust on blocks diagonally upward, either in a straight staircase of blocks, in a 2×2 spiral of blocks, or in another similar variation. Redstone staircases can transmit signals both upward and downward but can take up much space and require repeaters every 15 blocks.

#### Redstone ladders

Because glowstone, top slabs, glass, and upside-down stairs can support redstone dust but don't cut redstone dust, signals can be transmitted vertically (upward only) by alternating these blocks in a 2×1 "ladder". Redstone ladders take up less space than redstone staircases, but also require repeaters every 15 blocks. In *Bedrock Edition*, glass and pistons can be used to create two-way vertical ladders that transmit signals both upward and downward (glowstone, hoppers, and slabs still allow the dust to power upward but not downward).

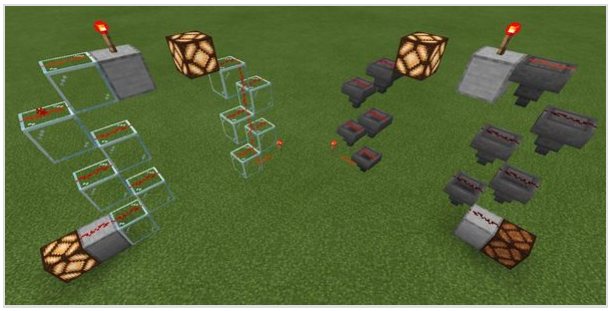
#### Torch towers and torch ladders

A redstone torch can power a block above it, or redstone dust beneath it, allowing vertical transmission both upward and downward (different designs are required for each). Because it takes each torch a little time to change state, a torch tower can introduce some delay into a circuit, but no repeaters are necessary. However, every torch inverts the redstone signal (i.e. changes it from powered to unpowered), so having an even number of torches is required.

#### Observer towers

An observer can power a block of a redstone circuit above or below it, allowing vertical transmission both upward and downward. Placing blocks that can be activated, such as redstone dust, note blocks or doors, both above and below it creates a state change to the block above when the observer is looking downward or

to the block below when the observer is looking upward. Repeating this pattern means that updates are chained.



Examples of vertical ladders in Bedrock Edition. Note how the glass facilitates transmission in both directions, but the hoppers allow signals to be sent only upward.

**Daylight detector exploiting**

You can use daylight detectors to send a Redstone signal downward in 1 tick, but the path needs to be unobstructed by anything. You need to have a piston push a block over the sensor. It detects the change in light and emits a Redstone pulse. This design is extendable upward as far as you want, but you need to have the original hole open to sunlight. It also works only during the day, because it uses shadows to activate.

**Bubble columns**

An observer can be used to detect the block update that occurs when a water source changes to a bubble column (or vice versa). When swapping the block below a column of water sources to soul sand or a magma block from some other block, the entire column immediately changes to bubble column blocks. This can be used to quickly transmit a redstone signal upward to an observer facing the top water source/ bubble column block.

**Wall updating**

A setup that can carry a pulse signal downward across any distance involves walls of any type, a piston, and an observer. When a wall block has a solid block on two opposing sides and non-solid blocks (e.g., air) on the other two sides, it takes a flat shape. This is vertically repeatable up to any height. However, when a wall/solid block is placed into one of the two air blocks around a flat wall, the flat wall block *and every flat wall block below it* are updated to a different version of the wall with a column in the middle. This update is instant and can be detected by an observer watching any flat wall in the tower. The update can be made repeatable by having a regular piston face the flat wall at the top of the tower, since the piston head also triggers the wall update.

**Repeater**

To "repeat" a signal means to boost it back up to full strength. The easiest way to do this is with a redstone repeater. Variations include:

**Instant repeater**

Repeats a solid signal without the delay introduced by a redstone repeater.

**Two-way repeater**

Repeats a signal in both directions.

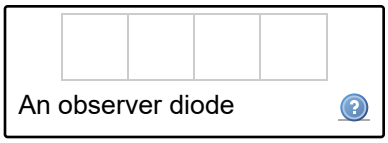
**Comparator**

A comparator has two modes: comparing and subtracting. Comparing mode compares the signal strength from the side with the input; if the side signal is stronger than the input signal, the output is off, otherwise it is on. In Subtracting mode, the comparator subtracts the side signal from the input.

**Diode**

A "diode" is a one-way circuit that allows a signal to travel in one direction. It is used to

protect another circuit from the chance of a signal trying to enter through the output, which could incorrectly change the circuit's state or interfere with its timing. It is also used in a compact circuit to keep one part of the circuit from interfering with another. Common choices for a diode include a redstone repeater or a height elevation to glass[*Java Edition only*] or a top slab, which does not transmit a signal back down. Many circuits are already one-way simply because their output comes from a block that can't take input. For example, a signal cannot be pushed back into a circuit through a redstone torch except through the block it's attached to.


An observer diode

## Logic circuit

*Main article: Logic circuit*

A **logic gate** is a circuit that takes one or more inputs and produces a single output when some condition has been met. The strength of the input signal in these circuits usually doesn't matter. When an input or output supplies a redstone signal, it is usually just referred to as "on", and when there is no signal, it is referred to as "off". For example, an AND gate takes two redstone signals as input, and outputs a redstone signal only if both inputs are greater than 0. That is, an AND gate is on only when both inputs are on.

In electronic or programming diagrams, logic gates are typically shown as if they were individual devices; However, when building redstone devices in *Minecraft*, all logic gates are formed from multiple blocks and components, which interact to produce the desired results.

**NOT gate**
: A NOT gate (aka "inverter") is on if its input is off. The simplest NOT gate is an input signal with a redstone torch attached.

**OR gate**
: An OR gate is on if *any* of its inputs are on. The simplest OR gate is to feed multiple input signals together into a single block or redstone wire.

**NOR gate**
: A NOR gate is on only if *none* of its inputs are on. The simplest NOR gate is to feed multiple input signals into a block with a redstone torch attached.

**AND gate**
: An AND gate is on only if *all* of its inputs are on. The simplest AND gate is to add redstone torches in front of each input to a NOR gate.

**NAND gate**
: A NAND gate is on if *any* of its inputs are off. The simplest NAND gate is to remove the final redstone torch from an AND gate.

**XOR gate**
: An XOR gate is on if its inputs are *different*.

**XNOR gate**
: An XNOR gate is on if its inputs are *equal*.

**IMPLY gate**
: An IMPLY gate is on unless the first input is on and the second input is off.

Logic gate outputs
Shows the output (red) of each gate, for each combination of inputs A and B (green).

| A | On | On | Off | Off | Question Answered |
|---|---|---|---|---|---|
| B | On | Off | On | Off | |
| A AND B | ON | Off | Off | Off | Are A and B on? |
| NOT (A IMPLIES B) | Off | ON | Off | Off | Is A on and B off? |
| NOT (B IMPLIES A) | Off | Off | ON | Off | Is B on and A off? |
| A NOR B | Off | Off | Off | ON | Are both inputs off? |
| A | ON | ON | Off | Off | Is A on? |
| A XOR B | Off | ON | ON | Off | Are the inputs different? |
| NOT A | Off | Off | ON | ON | Is A off? |
| A XNOR B | ON | Off | Off | ON | Are the inputs the same? |
| B | ON | Off | ON | Off | Is B on? |
| NOT B | Off | ON | Off | ON | Is B off? |
| A NAND B | Off | ON | ON | ON | Is either input off? |
| A IMPLIES B | ON | Off | ON | ON | If A is on, is B also on? |
| B IMPLIES A | ON | ON | Off | ON | If B is on, is A also on? |
| A OR B | ON | ON | ON | Off | Is either input on? |

## Pulse circuit

*Main article: Pulse circuit*

A **pulse circuit** produces a redstone signal for a specified amount of time, changes the duration of an input signal, or reacts to signals of a particular duration. Other circuits or mechanisms may require pulses of a particular duration to operate properly, or can use pulse duration to convey information.

A circuit that is stable in one output state and unstable in the other is known as a monostable circuit.[note 1] Many pulse circuits are monostable because their OFF state is stable, but their ON state soon reverts to OFF.

**Pulse generator**
: A pulse generator produces a pulse of a specific duration.

**Pulse limiter**
: A pulse limiter (aka pulse shortener) reduces the duration of pulses that are too long.

**Pulse extender**
: A pulse extender (aka pulse sustainer, pulse lengthener) increases the duration of pulses that are too short.

**Pulse multiplier**
: A pulse multiplier outputs multiple pulses for every input pulse (it multiplies the number of pulses).

**Pulse divider**
: A pulse divider (aka pulse counter) outputs a signal only after a certain number of pulses have been detected through the input (the number of pulses is indicative of the number of loops).

**Edge detector**
: An edge detector reacts to either a redstone signal changing from OFF to ON (a "rising edge" detector), from ON to OFF (a "falling edge" detector), or switching between ON and OFF in either order (a "dual edge" detector).

**Pulse length detector**
: A pulse length detector reacts only to pulses in a certain range of durations (often only to pulses of one specific duration).

## Clock circuit

*Main article: Clock circuit*

A clock circuit is a pulse generator that produces a loop of specific pulses repeatedly. Some are designed to run forever, while others can be stopped and started.

A simple clock with only two states of equal duration is named for the duration of its ON state (e.g., for example, a clock that alternates between a 5-tick ON state and a 5-tick OFF state is called a 5-clock) while others are usually named for their period (the time it takes for the clock to return to its original state; for example, a "1-minute clock" might produce a 1-tick pulse every 60 seconds).

**Observer clock 1**
: A repeating clock made with Observers and Pistons (an Observer looking at a piston).

**Observer clock 2**
: A repeating clock made with two Observers with their faces facing each other.

**Repeater clock**
: A repeater clock consists of a loop of repeaters (usually either redstone repeaters or redstone torches) with occasional dust or blocks to draw off the appropriate pulses.

**Hopper clock**
: A hopper clock produces timed pulses by moving items back and forth between 2 hoppers feeding into each other and taking a redstone output with comparators.

**Piston clock**
: A piston clock produces a loop of pulses by passing a block back and forth (or around, with many pistons) and drawing off a redstone pulse when the block is in a certain location.

**Comparator clock**
: The clock of short or moderate cycle length utilizing comparator's subtraction or signal fading feature. Clocks can also be built using daylight sensors, minecarts, boats, water flow, item despawn, etc.

## Memory circuit

*Main article: Memory circuit*

Memory circuits maintain their output until they receive a signal that tells them to change their output. In simple terms, memory circuits stay on or off until they receive a signal telling them to turn off or on. This means that a memory circuit's state (output) depends on the current input

and the previous inputs. Most other circuits just temporarily generate an output based on the current input.

Most memory circuits are called "latches" or "flip-flops"; these are named after real-life electronic circuits because they behave similarly. Some basic types of memory circuits are outlined below. See here for an explanation of technical terms.

Memory circuits are named based on the kinds of input they take and any logic gates they use.

**RS latch**
: An RS latch has two inputs: set (S) that turns the circuit ON, and reset (R) that turns the circuit OFF. The oldest and most common memory circuit in *Minecraft* is the RS latch built with a NOR gate (RS NOR latch).

**T flip-flop**
: A T flip-flop has one input: toggle (T) that changes the circuit's output from ON to OFF, or from OFF to ON.

**Gated D latch**
: A gated D latch has two inputs: clock (C) that determines when the circuit is allowed to change its output, and data (D) that determines the output. It is called "gated", because the clock acts as a gate, allowing the circuit to update depending on the clock state.

**JK latch**
: A JK latch has two inputs: J, which turns the circuit ON, K which turns the circuit OFF; if J and K are both ON, then the circuit toggles its output. These circuits also often have a clock (C) input that determines when the circuit is allowed to change its output. (J and K do not have common names like set, reset, toggle, etc. do).

**Counter**
: Unlike T flip-flops and RS latches, which can hold two states (ON or OFF), a counter can be designed to hold a greater number of states.

Many other memory circuits are possible.

## Piston circuits

*Main article: Piston circuits*

Pistons allow players to design circuits that are smaller and/or faster than the standard, redstone-only counterparts. An understanding of standard redstone circuits is helpful, as this tutorial is focused on the circuit design rather than the function. The main components here are sticky pistons, regular pistons, redstone wire, repeaters and redstone torches.

There are several benefits of piston circuitry:

- Neither repeaters nor pistons 'burn out', unlike redstone torches.
- Piston circuits are often (not always) smaller and/or faster than their redstone counterparts. This allows building devices such as fast clocks and "instant" signal transmission.
- Pistons' ability to move blocks within the world makes them a natural for memory circuits, as well as the obvious doorways and switchable bridges. With slime or honey blocks involved, entire structures can "get up and move" (see also tutorial on flying machines).
- Piston circuits can sharply reduce the use of redstone in favor of wood, stone, and iron.

## Miscellaneous circuits

*Main article: Miscellaneous circuits*

These circuits aren't generally needed for redstone projects, but might find use in complex projects, proofs of concept, and thought experiments. Some examples:

**Multiplexers and relays**
A multiplexer is an advanced form of logic gate that chooses which of two inputs to let through as output based on an additional input (for example, if input A is ON then output input B, otherwise output input C). The reverse of this is a relay, which copies a data input to one of two outputs, depending on whether the additional input is ON or OFF.

**Randomizers**
A randomizer produces output signals unpredictably. Randomizers can be designed to produce a pulse at random intervals, or to randomize which of multiple outputs are turned ON (such as random number generators, or RNGs). Some randomizers use the random nature of *Minecraft* (such as cactus growth or dispenser slot selection), while others produce pseudo-randomness algorithmically.

*Main article: Tutorial:Randomizers*

**Multi-bit circuits**
Multi-bit circuits treat their input lines as a single multi-bit value (something other than zero and one) and perform an operation on them all at once. With such circuits, possibly combined with arrays of memory circuits, it's possible to build calculators, digital clocks, and even basic computers inside *Minecraft*.

**Block update detectors**
A block update detector (BUD, or BUD switch) is a circuit that reacts to block updates (specifically neighbor updates in *Java Edition*). BUDs react by producing a pulse, while T-BUDs (toggleable BUDs) react by toggling their output state. These are generally based on subtle quirks or glitches in device behavior; current circuits most often depend on pistons. Many of the functions of BUDs can also be replicated with observers, but since observers detect a different kind of block updates,[*JE only*] using BUDs instead might be necessary in some situations.

*Main articles: Tutorial:Block update detector and Tutorial:Comparator update detector*

**More advanced circuits**
Many other complex circuits are possible.

*Main article: Tutorial:Advanced redstone circuits*

# Achievements

[hide]

| Icon | | Achievement | In-game description | Actual requirements (if different) | Gamerscore earned | Trophy type (PS) |
|---|---|---|---|---|---|---|
| PS4 | Other | | | | | |
|  |  | On A Rail | Travel by minecart to a point at least 500m in a single direction from where you started. | Travel by minecart 500 blocks in a straight line away from the player's starting point. | 40 | Gold |
|  |  | Inception | Push a piston with a piston, then pull the original piston with that piston. | — | 20 | Silver |
|  |  | Crafters Crafting Crafters | Be near a Crafter when it crafts a Crafter | — | 10 | Bronze |

## Videos

## References

1. Note: Some players refer to edge detectors as monostable circuits

## Navigation

**Redstone** [hide]

**Redstone circuits & tutorials** [hide]

| | |
|---|---|
| **Redstone basics** | Redstone components · Redstone mechanics · Conductivity |
| **Redstone circuits** | Clock circuits · Logic circuits · Memory circuits · Miscellaneous circuits · Pulse circuits · Transmission circuits |
| **Featured tutorials** | Advanced redstone circuits · Block update detector (BUD) · Comparator update detector (CUD) · Flying machine · Mechanisms · Piston circuits · Quasi-connectivity · Redstone music |

**Redstone components** [show]

**Gameplay** [hide]

| | |
|---|---|
| **General mechanics** | Add-ons · Attribute · Commands · Distance · Effect · Explosion · Game rules · Interaction range · Inventory (Creative inventory (Saved Hotbars)) · Generated loot · Hitbox · Multiplayer (Servers · Server list · Realms · Splitscreen) · Oxidation · Rarity (Legacy) · Redstone circuits (Conductivity) · Rotation · Snowlogging · Social · South-east rule · Spawn protection · Tiers · Vibration · Waterlogging |
| **Technical mechanics** | Anvil mechanics (Legacy) · Black entities · Enchanting table mechanics · Redstone mechanics · Village mechanics (Legacy) |
| **Survival** | Achievements · Advancements · Archaeology · Armor materials · Bartering · Breaking (Instant mining) · Breeding · Brewing · Cooking · Crafting (2×2 grid · Recipe book) · Death · Dual wield · Difficulty · Durability · Enchanting · Experience · Farming |

| | |
|---|---|
| **Combat** | Fishing Health Healing Food mechanics (Hunger Saturation) Item repair Mob conversion Mob spawning Mob types Ominous Event (Ominous Trial Raid) Patrol Raid captain Renewability (Renewable Non-renewable) Smelting Smithing World spawn Taming Trading Workstations Zombie siege<br>Damage Knockback Melee attack (Attack damage Attack cooldown Attack range Special attack) Ranged attack (Projectile damage Charge time) Shield blocking Drops Mob infighting Geared mobs Mob fleeing Use cooldown<br>**More** |
| **Environment** | Biomes Daylight cycle Dimensions Seeds Structures Weather World generation<br>**More** |
| **Movement** | Crawling Flying Gliding Jumping Lying Riding Sitting Sneaking Sprinting Swimming Teleportation Walking |
| **User interface** | Action bar Bossbar Chat (Death messages) Font Game mode switcher Heads-up display Language Narrator Locator Bar Scoreboard Toasts (Tutorial hints) Tooltip |
| **Visuals** | Color (Block colors Item colors Effect colors Miscellaneous colors) Emotes Enchantment glint Error Light Resource pack Screen effects Skins (Character Skin pack) Third-person view Vibrant Visuals |
| **Removed** | Sword blocking Materials |
| **Unintended** | Update suppression Duplication |