

Redstone mechanics 🗨 [re article feedback](#)

Redstone mechanics describe the general concepts of redstone power, and how redstone-related blocks interact with one another. This article describes basic ideas such as redstone power, powering blocks, and activating redstone mechanisms.

Redstone is *Minecraft's* loose analog to electricity. There are a variety of blocks that are able to create a redstone signal or perform unique actions when provided with a redstone signal. Redstone-related blocks can be used in combination to create [circuits](#) and complicated [mechanisms](#) that can be used to make automatic farms, item sorters, flying machines, and more.

Contents

Redstone concepts

Terminology

Redstone tick

Conductive and non-conductive blocks

Strongly powered vs. weakly powered

Redstone components

Redstone signal

Signal generation

Signal transmission

Activating mechanisms

Activation behavior

Quasi-connectivity

Powered blocks

Strongly powered

Weakly powered

Powered vs. activated

Conductivity

Redstone block updates

Navigation

Redstone concepts

Terminology

The *Minecraft* community has created a variety of terms to describe different aspects of redstone behavior, either for convenience or because there is no official term to describe the behavior. Terminology regarding redstone is not uniform throughout the community. A few of these terms and their variations are detailed below.

Redstone tick

Main article: [Tick § Redstone tick](#)

A game tick is the basic unit of time in *Minecraft*, and is equivalent to $\frac{1}{20}$ seconds (0.05 seconds). Every game tick, various aspects of the game are updated such as player movement, block updates, mob spawns, etc.

In *Java Edition*, most (but not all) redstone-related events occur in multiples of 2 game ticks. A common convention is to measure times of redstone events in "redstone ticks" which are equivalent to 2 game ticks (0.1 seconds). Events that happen in 1 game tick are said to occur in 0.5 redstone ticks. "Redstone tick" is a community-created term, and not an official unit of time in *Java Minecraft*. The *Minecraft* game code and commands such as `/tick` use game ticks.

In *Bedrock Edition*, a redstone tick is a unit of time that describes a delay of two game ticks, creating a $\frac{1}{10}$ of a second delay.

The redstone system is calculated on concurrent threads, meaning that redstone updates are processed independently of the main game loop. This architecture utilizes a two-phase tick system, known as produce ticks (P-Ticks) and consume ticks (C-ticks), or alternatively as Output Ticks and Input Ticks.

These two phases form a single redstone tick, which spans 2 game ticks. There are 10 redstone ticks per standard 20-game-tick second.

Tick cycle

- Output/produce tick (P-tick): Occurs first on odd-numbered game ticks. During this phase, components output or send their signal.
- Input/consume tick (C-tick): Occurs second on even-numbered game ticks. During this phase, components read or react to signals they are receiving.

Component Behavior

- Powerable blocks (pistons, redstone lamps, doors, etc.): These components are activated during an Input/consume tick and change their block state, which then creates block updates processed in the subsequent Output/produce tick.
- Repeaters, comparators, redstone torches, and observers: These components read their input side during an output/produce tick. After their configured delay, they update their output side and produce a input/consume tick.
- Redstone dust updates its power state on every game tick (20 times per second), independently of the dedicated redstone tick system. This allows it to transmit pulses with a duration of only 1 game tick. A dust line powered on an odd game tick serves as an output for components processing on P-Ticks, while one powered on an even game tick serves as an output for components on C-Ticks. This makes redstone dust one of the few components

that can be "1-ticked" and used to create phase-specific pulses. Redstone dust only updates visually on output/produce ticks, meaning on input/consume tick while technically on looks visually off.

- Armor stands update on both input and output ticks meaning they are one of the few redstone components that utilizes all 20 ticks.

The transmission and timing of redstone signals are determined by these ticks, which govern when redstone devices output power and when they respond to input.

Conductive and non-conductive blocks

Conductivity is the idea that some blocks can be powered and some cannot. Conductive blocks are blocks that can be powered and non-conductive blocks are blocks that cannot be powered.

Conductive blocks are often called "opaque" blocks, because many common blocks that can be powered are opaque; for example, stone, dirt, wood. Non-conductive blocks are often called "transparent" blocks, because the most common example is glass. However, whether or not you can see through the block does not determine whether a block can be powered.

Strongly powered vs. weakly powered

When some blocks receive a redstone signal, they can activate redstone mechanisms or provide a redstone signal to certain blocks. These blocks are said to be "powered".

Strongly powered blocks can power redstone dust, and weakly powered blocks cannot. Strongly powered is sometimes called "hard powered", and weakly powered is sometimes called "soft powered".

Redstone components

Main article: Redstone components

Redstone components are blocks that are used to create redstone circuits and advanced mechanisms.

- A **power component** provides a redstone signal. Some power components can produce a pulsed or continuous signal on demand. For example, a button or lever. Others power components produce a pulsed or continuous signal when specific conditions have been met. For example, a pressure plate, or an observer.
- A **transmission component** carries a redstone signal from one part of a circuit to another. Redstone dust is the most basic transmission component; redstone repeaters and redstone comparators are also transmission components.
- A **mechanism component** performs an action when it receives a redstone signal. For example, a piston extends and pushes a block, or a redstone lamp turns on and acts as a light source.

Redstone signal

A redstone signal is *Minecraft's* analog to electricity, and is generated by redstone power components. A redstone signal has a "signal strength", which is an integer between 1 and 15. The strength of a signal does not affect how redstone mechanism components operate. For example, a piston extends the same length and at the same speed regardless of whether it receives a redstone signal of 1 or 15. A redstone lamp produces the same amount of light, no matter the strength of the signal.

Signal generation

Only redstone power components can generate a redstone signal. Some components generate a continuous signal, while others generate a signal pulse. Some components generate a signal on demand, such as a lever or a button. Other components generate a signal when some condition has been met, such as a mob walking over a pressure plate, or an observer detecting a block change.

Sculk sensors and "wireless" redstone

Sculk sensors and their variant, calibrated sculk sensors, can be used to generate a "wireless" redstone signal. Sculk sensors cannot actually transmit a redstone signal across distances, but can be used to generate a redstone signal remotely.

When a sculk sensor detects a vibration, an attached redstone comparator outputs a specific redstone signal that depends on the cause of the vibration. Sculk sensors can be set up to retransmit a detected vibration through the air to another sculk sensor. The player can create a specific vibration at their location, and have that vibration be retransmitted across a chain of sculk sensors over a long distance. The final sculk sensor in the chain provides the redstone signal through a redstone comparator.

Signal transmission

A redstone signal can be transmitted only by redstone transmission components such as redstone dust, redstone repeaters, and redstone comparators.

Redstone dust

Redstone dust is used to transmit a redstone signal, and is analogous to a wire that can carry electricity. Redstone dust transmits power to adjacent redstone dust, but the signal strength decreases by 1 for every block of redstone dust that the signal travels. Redstone dust can thus transmit a signal up to 15 blocks by itself. The signal can be extended farther using other transmission components such as a redstone comparator, or a redstone repeater.

Signal strength does not decrease when transmitted from redstone dust to a block or a redstone component. For example, redstone dust carrying a signal of strength 1 and pointing into a piston causes the piston to extend. If redstone dust with a signal strength of 1 points into a conductive block, a redstone repeater or a redstone comparator facing away from the block is powered.

Redstone repeater

When a redstone repeater receives a redstone signal of any strength, it outputs a signal of strength 15.

Redstone comparator

When a redstone comparator receives a signal from its back input, it outputs the same signal.

Activating mechanisms

Redstone mechanism components are activated when they receive a redstone signal. This signal can be supplied by an adjacent power component generating a signal, a powered block, powered redstone dust being configured to point into the mechanism or placed on top of the mechanism, or a powered redstone repeater or a redstone comparator pointing into the mechanism. A mechanism component performs some kind of action when activated, such as moving, producing light, or making a sound.



Activation of Mechanism Components —

Mechanism components can be activated by power components (for example, redstone torches), powered blocks, redstone dust, repeaters, and comparators (not shown), but only if configured correctly.

Activation behavior

Upon receiving a redstone signal, mechanism components may behave in different ways. Some mechanisms perform an action only once when they receive a signal, others continuously perform an action while receiving a redstone signal.

Rising edge

Some mechanism components perform an action only once upon receiving a redstone signal. These components do not perform another action until the signal stops, and they receive a new signal some time later. This type of behavior is referred to as rising edge because they occur when the signal goes from 0 then up to a positive value. The opposite is falling edge, where a signal goes from a positive value to 0.

Rising and falling edge

Some mechanism components change state when they receive a redstone signal (rising edge activation) and return to their original state when the redstone signal stops (falling edge deactivation). These blocks have an initial deactivated state, and an activated state. For example, a redstone lamp starts producing light when it receives a redstone signal, and does not stop until it stops receiving the signal.

For blocks such as doors, fence gates, and trap doors, their deactivated state is closed, and their activated state is open. If one of these blocks is already open when it receives a redstone signal, it stays open, but closes when the signal stops.

Mixed Behavior

A copper bulb exhibits mixed behavior. A copper bulb has two block states associated with it: lit, describing if the copper bulb is emitting light, and powered, describing if the copper bulb is currently receiving a redstone signal. Upon receiving a redstone signal (rising edge), the bulb toggles its lit state, and starts or stops emitting light. The bulb's powered state remains true until it stops receiving a redstone signal (falling edge).

Quasi-connectivity

Main article: [Tutorial:Quasi-connectivity](#)



This feature is exclusive to [Java Edition](#).

Almost every mechanism component block needs a redstone signal to be supplied to the block itself in order to be activated. There are three exceptions in [Java Edition](#): **dispensers**, **droppers**, and **pistons**. These three blocks can be activated by a redstone signal being supplied to them like other blocks, but can also be activated if a redstone signal is supplied to the block above them, even if that block is air.

Another way to think of quasi-connectivity, is to imagine that dispensers, droppers, and pistons have an "activation hitbox" similar to that of a door. A door is two blocks tall, and if either the top or bottom part of a door receives a redstone signal, it opens. Dispensers, droppers, and pistons are one block tall, but they can be activated as if they were two blocks tall.

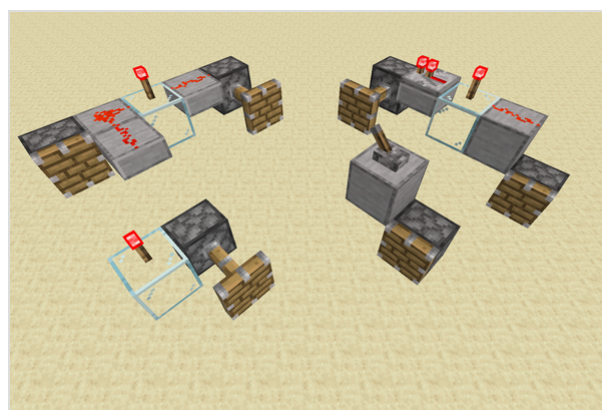
A complication of quasi-connectivity is that there are scenarios in which, the dispenser, dropper, or piston should be activated by quasi-connectivity, but does not activate until there is a nearby block update. This occurs because redstone blocks can update their neighbors up to two blocks away, [taxicab distance](#). Even though the dispensers, droppers, and pistons have a two block tall activation hit box, the actual block needs to be updated in order to activate, not the block above. Sometimes the source of the redstone signal is too far away to update the actual block. See main article for additional details.

Powered blocks

"Powered" redirects here. For the enchantment with a similar name, see [Power](#).

When [conductive](#) blocks, often referred to as *opaque* blocks ([stone](#), [dirt](#), etc.), receive a redstone signal, they can activate mechanism components or power transmission components.

A block can be powered by:



Activation by [Quasi-Connectivity](#) — Pistons can also be activated by anything that activates the space *above* them. Note that the piston on the far left is *not* activated by quasi-connectivity because the redstone dust is running *past* the block above the piston, rather than directly into it, and thus would not power a mechanism there

- a powered redstone dust pointing into, or laying on top of a block
- a powered redstone repeater, or redstone comparator pointing into a block
- a redstone power component that is capable of powering a block (see list of [power components](#) for details)

A powered block can activate adjacent redstone mechanism components, and power redstone repeaters and comparators facing away from the block. A powered block may or may not be able to power adjacent redstone dust, depending if it has been strongly or weakly powered.

Blocks that cannot be powered are usually referred to as transparent blocks, [glass](#) being a common example.

Strongly powered

A block is strongly powered when it can power adjacent redstone dust (including redstone dust on and beneath the block), in addition to activating adjacent mechanism component, and powering redstone repeaters and redstone comparators facing away from the block. A block becomes strongly powered by being powered by a redstone power component, a powered redstone repeater, or a powered redstone comparator.

Weakly powered

A block that is weakly powered *cannot* power adjacent redstone dust, but can still activate adjacent redstone mechanisms, and power redstone repeaters and redstone comparators facing away from the block. A block becomes weakly powered when it is powered only by redstone dust.

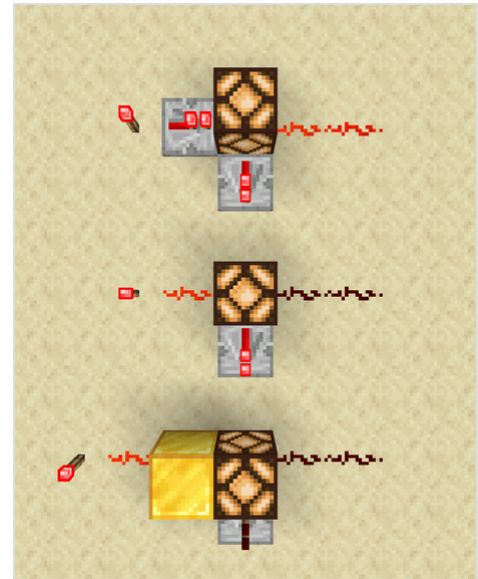
Powered vs. activated

Some redstone mechanism components, such as redstone lamps, droppers, and dispensers are [conductive](#) blocks and can be powered. When one of these blocks is powered, it is simultaneously activated (excluding copper bulbs, which can be powered and inactive). These blocks do not need to be powered in order to be activated. If one of these blocks is activated, it does not mean that it is powered.

Conductivity

Main article: [Conductivity](#)

Conductivity is a [block property](#) that determines which blocks can be [powered](#) and which cannot.



The Redstone Lamps are all **activated**, but are **powered** differently. From top to bottom:

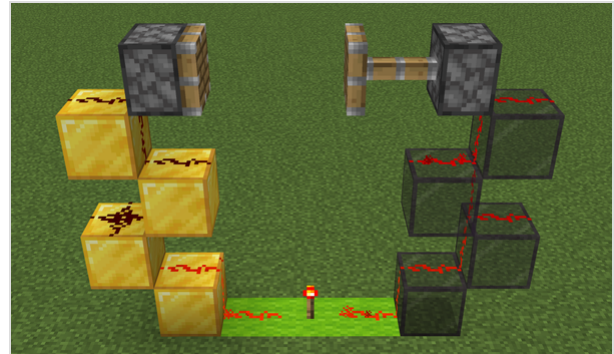
1. Strongly powered: powers both Repeater and Dust.
2. Weakly powered: powers Repeater, but not Dust.
3. Not powered: powers neither.

Specifically, conductive blocks can be powered and non-conductive blocks cannot be powered. Conductive blocks are often called "opaque" blocks, because many common blocks that can be powered are opaque; for example, stone, dirt, wood. Non-conductive blocks are often called "transparent" blocks, because the most common example is glass. However, whether or not you can see through the block does not determine whether a block can be powered. Most conductive blocks share a similar set of properties, and most non-conductive blocks share their own set.

Most conductive blocks are made of materials that have the solid-blocking property, and are full blocks, with exceptions noted below.

Common properties of conductive blocks

- can be powered
- prevent chests from opening when placed on top of them
- suffocates mobs inside them
- can "cut" redstone dust (prevent redstone dust from connecting vertically, see image)
- redstone dust can transmit a signal up or down a conductive block
- comparators can measure block state through a conductive block



The conductive gold block cuts the redstone dust, preventing it from connecting vertically and does not transmit a signal; the non-conductive tinted glass does not cut the redstone dust, allowing it to connect vertically and transmit a signal

Common properties of non-conductive blocks

- cannot be powered
- do not prevent chest from opening when placed on top of them
- do not suffocate mobs inside them
- do not cut redstone dust (prevent from connecting vertically)
- redstone dust can transmit a signal only up a non-conductive block, not down
- comparators cannot measure block state through non-conductive blocks

Unique conductive blocks

As noted above, most conductive block are full solid blocks with the exception of soul sand and mud (7/8 of a block tall).

Mangrove roots, Monster spawners^[*Bedrock Edition only*], Trial spawners^[*Bedrock Edition only*] and closed shulker boxes^[*Bedrock Edition only*] are the only conductive block that can be waterlogged.







Unique non-conductive blocks

Blocks of redstone, observers, and pistons are full solid blocks made of materials with the solid-blocking property, but are non-conductive.

Non-conductive block placement behavior

Many non-conductive blocks exhibit their own unique behavior regarding what blocks can be placed on them, and what blocks they can be placed on. See Opacity/Placement.

For example, leaves are a full solid non-conductive block, but certain redstone-related blocks cannot be placed on them:

-  redstone dust
-  redstone repeater
-  redstone comparator
-  button
-  lever
-  tripwire hook
-  rail, and its varieties

Redstone block updates

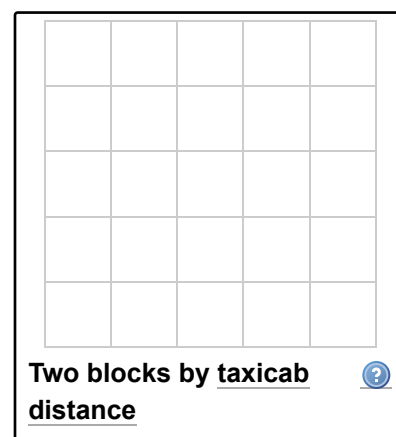
Main article: Block update

Block updates are how most redstone components "tell" each other that they need to change states.

When a change occurs somewhere in a redstone circuit, it can produce other changes in surrounding blocks in what is called a **block update**. Each of these changes can then produce other changes in their surrounding blocks. The update propagates following the redstone circuit rules within loaded chunks (block updates do not propagate into unloaded chunks), usually quickly. *Note: In Bedrock Edition, block updates and redstone are not connected.*

A block update simply notifies other Redstone components and blocks that a change has occurred nearby and allows them to change their own state in response, but not all updates necessarily require changes. For example, if a redstone torch activates and updates the dust below it, the dust may already be powered from something else, in which case the dust doesn't change state and the update propagation stops there.



Block updates can also be generated by any immediate neighbor block being placed, moved, or destroyed.













Solid blocks don't "know" if they're powered or not. Block updates simply update enough blocks around a redstone component to update other redstone components around the solid block (for example, a pressure plate updates its neighbors and the neighbors of the block it's attached to, including the space under that block, which might be redstone dust).

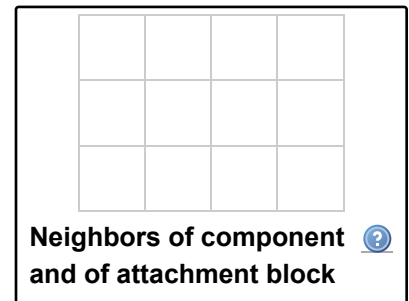
In addition to block updates, comparators can be updated by containers (including detector rails with container minecarts on them) and certain other blocks, up to two blocks away horizontally when their state changes (for example, when their inventory changes). This is known as a comparator update.

The following redstone components produce block updates up to two blocks away by taxicab distance, including up and down:







-  Redstone Dust (All directions)
-  Redstone Torch (Up and down)
- Flat and slanted rails, activator rails, detector rails, and powered rails (up and down if slanted, or only down if flat)

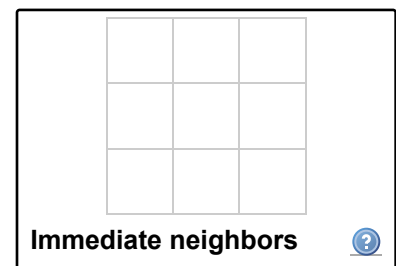
The following redstone components produce block updates in their immediate neighbors, including above and below, *and* in the immediate neighbors of the block they're attached to:

-  Redstone Repeater (as if "attached" to the block it is facing)
-  Redstone Comparator (as if "attached" to the block it is facing)
-  Buttons
-  Detector Rail (flat only; also produces comparator updates)
-  Lever
-  Pressure Plates
-  Trapped Chest (as if "attached" to the block beneath; also produces comparator updates)
-  Tripwire Hook
-  Weighted Pressure Plates
-  Observer













The following redstone components update only their immediate neighbors when they change their state, including above and below:

-  Daylight Detector
-  Inverted Daylight Detector (the Inverted Daylight Detector is not obtainable as an item)
-  Note Block
-  Leaves
-  Scaffolding
-  Tripwire (can also activate tripwire hooks in valid tripwire circuit)





































































-  [Piston](#) and [Sticky Piston](#) (from both the piston base and the piston head when extended)

The following redstone components do *not* produce block updates when they change their state (although any block produces a block update in its immediate neighbors if moved or destroyed):

-  [Impulse Command Block](#) (also produces comparator updates)
-  [Repeating Command Block](#) (also produces comparator updates)
-  [Chain Command Block](#) (also produces comparator updates)
-  [Dispenser](#) (also produces comparator updates)
-  [Dropper](#) (also produces comparator updates)
-  [Doors](#)
-  [Fence Gates](#) (can be moved)
-  [Hopper](#) (also produces comparator updates)
-  [Redstone Lamp](#) (can be moved)
-  [Trapdoors](#) (can be moved)

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 |
| |  Advanced redstone circuits  Block update detector (BUD) |
| Featured tutorials |  Comparator update detector (CUD)  Flying machine  Mechanisms  Piston circuits  Quasi-connectivity  Redstone music |
| | Redstone components [show] |
| | Gameplay [hide] |
| |  Add-ons  Attribute  Commands  Distance  Effect |
| |  Explosion  Game rules Interaction range  Inventory |
| | ( Creative inventory ( Saved Hotbars))  Generated loot  Hitbox |
| General mechanics |  Multiplayer ( Servers  Server list  Realms  Splitscreen)  Oxidation  Rarity (Legacy)  Redstone circuits ( Conductivity)  Rotation  Snowlogging |
| |  Social  South-east rule  Spawn protection  Tiers |
| |  Vibration  Waterlogging |
| |  Anvil mechanics (Legacy)  Black entities |
| Technical mechanics |  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 |

| | |
|-----------------------|--|
| | 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 |
| | Damage Knockback Melee attack Attack damage |
| | Attack cooldown Attack range Special attack |
| | Ranged attack Projectile damage Charge time |
| Combat | Shield blocking Drops Mob infighting Geared mobs |
| | Mob fleeing Use cooldown |
| | More |
| | Biomes Daylight cycle Dimensions Seeds |
| Environment | Structures Weather World generation |
| | More |
| | Crawling Flying Gliding Jumping Lying Riding |
| Movement | Sitting Sneaking Sprinting Swimming |
| | Teleportation Walking |
| | Action bar Bossbar Chat Death messages Font |
| User interface | Game mode switcher Heads-up display Language |
| | Narrator Locator Bar Scoreboard Toasts |
| | Tutorial hints Tooltip |
| | Color Block colors Item colors Effect colors |
| Visuals | 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 |

Retrieved from "https://minecraft.wiki/w/Redstone_mechanics?oldid=3343914"

This page was last edited on 1 January 2026, at 20:21.

Content on this site is licensed under CC BY-NC-SA 3.0 unless otherwise noted; additional terms may apply.

Not an official Minecraft website. We are not associated with Mojang or Microsoft.