Differences Between Minecraft: Bedrock Edition And Minecraft: Java Edition

There are two major versions of Minecraft: Bedrock Edition and Java Edition. On the surface, they may seem very similar, but under the hood is a completely different story. The different code bases create distinct development environments for content creators. This tutorial outlines the major differences you as a content creator should be aware of.

In this tutorial you will learn the following:

- A brief history of Java Edition and Bedrock Edition. - How the two editions differ and what it means for content creation.

This version was originally released in 2009. This version used to be called Minecraft until it was renamed to Minecraft: Java Edition in September 2017. As the name implies, it's developed in Java and isn't compatible with the current version of Minecraft for the most part. This edition is commonly referred to as simply "Java".

Minecraft: Bedrock Edition

Bedrock Edition was launched on September 20, 2017 and was based on Minecraft: Pocket Edition, which was released in 2011. It brought together nine of the major device platforms under a singular codebase called the Bedrock Engine. This was a rewrite of Minecraft from the ground up and brought along with it some fundamental changes to the platform paving the way for an exciting new development community. This edition is commonly referred to as simply "Bedrock".

World Format Differences

One major technical difference between both versions is the world format. Bedrock Edition uses the LevelDB format for world storage while Java Edition uses the Anvil format. Due to this, most third-party tools created for Java Edition world editing will not work on Bedrock Edition.

The two versions also use a fairly different block format. Java Edition has flattened its block format using a unique string for each individual block and storing the state of that block separately. Similarly, Bedrock Edition has moved to a string-based system with block states, but have kept some blocks grouped together defined by data value. Basically, this means that blocks are named differently between the versions. In Bedrock Edition, granite would be stone 1 whereas on Java Edition it's simply granite.

Redstone and Command Differences

The structure and implementation of commands between the two versions have diverged as well. Bedrock Edition's command structure is similar to the system used in versions of Java Edition prior to 1.13. It also forgoes raw JSON strings inside commands for a component-based system. Instead of using long complex JSON strings to customize entities, you can summon an entity with an event to fire, and also name it in a single command.

Redstone functions slightly different as well. Unlike Java Edition, Bedrock Edition doesn't support quasi-connectivity. Systems that utilize mechanics such as Block Update Detector (BUD) switches won't work. Pistons also require one tick to retract, and won't leave blocks behind if given a one-tick pulse. Even the way updates happen is slightly different. While the vast majority of redstone circuits work well between the two versions, more complex circuits might not.

Resource Packs

The idea behind resource packs is the same in both editions: change how the game looks. However, the capabilities and layout are very different.

Behavior Packs

Bedrock Edition's equivalent to Java's data packs are called behavior packs. Again, both share some similarities and some differences. Only Java Edition can change the shape of blocks, and only Bedrock Edition can change the shape of entities. Each uses a geometry format for this purpose that's incompatible with the other edition. To animate textures, Bedrock uses a single file called flipbook_textures.json, while Java uses individual .mcmeta files for each texture. Java Edition can create custom fonts and GLSL shaders, while Bedrock cannot. Bedrock Edition can create custom particles and fogs, while Java cannot.

Gameplay and Player Input

One major difference that tends to be forgotten is the type of platform players of different versions use. For Java Edition, you can be reasonably sure your player is using a keyboard and mouse; on Bedrock Edition, more than likely your player is using console controls, with touch being a close second. Keyboard and mouse controls are a far third and make up a tiny percentage of your player base.

That means when designing experiences in Bedrock Edition, you should be aware of the different types of input players will be using. Also, keep in mind how your players are playing. While spam clicking might be OK with a mouse or even a controller, it would provide a poor experience for touch players. Keyboard players with a bow might have perfect aim, but it's a lot more difficult when using a controller or touch controls. Complex parkour might even be game-breaking for a mobile player.

Considerations

That means when designing experiences in Bedrock Edition, you should be aware of the different types of input players will be using and what their experience might be like on PC and console. Motorsnmore Also, keep in mind how your players are playing. While spam-clicking might be OK with a mouse or even a controller, it would provide a poor experience for touch players. Keyboard players with a bow might have perfect aim, but it's a lot more difficult when using a controller or touch controls. Complex parkour might even be game-breaking for a mobile player.

Besides normal bugs causing issues, the advanced features the platform provides also means there are more ways to break the game. Lots of entities with complicated behaviors can slow down some devices. Custom entities that use overly complex models can eat up RAM. Even the amount of chunks that can be loaded at once may be dramatically less on lower-end devices like mobile.

What's Next?

If you are coming newly from Java Edition, your first steps in Bedrock Edition will be Add-On development. This will open many doors necessary for content creation on Bedrock.