

# Upgrades

## What are Upgrades?

Upgrades allow players to enhance functionality of the map. They can be used to optimize existing systems, unlock new mechanics, or customize the game world.

## How to get Upgrades?

Getting/unlocking upgrades depends on the specific gamemode you are currently playing. Some gamemodes have ways to get upgrades, some are always on a specific upgrade level for certain upgradable things.

For example, in Conquest there are upgrade missions. Completing those missions will unlock an upgrade. In Rush, there is no way to upgrade the castle. The castle there is in a "fixed state", depending on the zone the gamemode is currently in.

## Upgrades

The upgrades on the default map are the following:

- Battery Upgrade: Increases the battery capacity and therefore the time the castle can run its electrical systems without an external power source.
- Status monitor upgrade: Adds new features like player detection to the status monitor.
- Door Upgrade: Adds centralized control of all doors in the castle through the door control monitor. On the first level, the status can be viewed. On the second, they can be opened or closed and on the third level they can be locked or unlocked.
- Camera Upgrade: Adds cameras to the castle to observe the enemies.

## Developer Info

## Control command

```
/cc2 map value upgrades [upgrade id] [level](null)]
```

# Responsibilities

Upgrades ( `CastleUpgrade` ) are managed by the `CastleUpgradeManager` ).

The `UpgradeAppliedEvent` is fired when an upgrade has been changed. It contains the previous and the new upgrade level and the upgrade id.

## How upgrades work

### Applying/removing Upgrades

The `CastleUpgrade` contains a list of `Level` interfaces (which are representing upgrade levels).

The `Level` interface contains an `onApply` and `onRemove` method. When an upgrade level has been changed, the `onRemove`-method of the old upgrade level is called. Then, the `onApply`-method of the new upgrade level is called. If the upgrade is removed (unlikely, but possible), only the `onRemove`-method of the previous applied level is called, without calling any `onApply` method.

In the implementation of the `onRemove` method, the entire status of the `Level` must be completely reversed. After `onRemove` has been called, none of the changes to the level may remain. **It is absolutely essential that a level is completely cleaned up in the `onRemove` method. No remnants of the old state may remain. If there are overlaps between the old and new level, these must be explicitly restored in `onApply` of the new level. IF YOU DON'T FOLLOW THIS RULE, THINGS MAY BREAK!**

### Example

When the `onRemove`-method of the battery upgrade is called, it completely removes the battery block structure, regardless if it is there or not. Then, in the `onApply`-method, it puts the new block structure into the world.

This ensures, that if the battery upgrade is not upgraded linear, but set to a random level, everything is set correctly. This also ensures that if the battery upgrade is removed, all batteries disappear.

---

Revision #1

Created 19 March 2025 20:15:48 by jandie1505

Updated 19 March 2025 20:16:52 by jandie1505