**A pixel art of a person walking

Description automatically generatedGabriel’s guide on How to Create a new Passive Item**

Passive Items give buffs to the player and the structures, to create a new one, you will need to create its icon and program its behaviour. I put some systems in place to make the process very straightforward and easy to integrate with the game.

A screenshot of a computer program

Description automatically generated

Every PassiveItem Asset has the attributes: Name, description, icon, prefab, probability and a list of levels. Each level has Effect intensity and a little description.

Every passive item needs to extend from the “PassiveItem” class. The base class has attributes such as references to its item asset and current level. The base class has overridable methods to apply buffs.

A screenshot of a computer

Description automatically generatedThe Player script has a “PlayerStats” class reference that contains many attributes that can be modified by passive items.

**How does it work?**

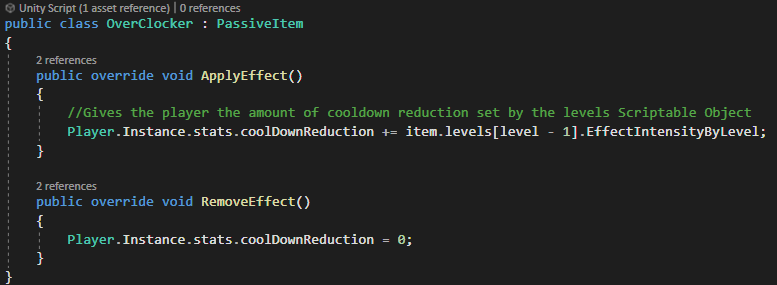
**A computer screen shot of a program

Description automatically generated**

Every time a structure is placed or a passive item is added, the “ApplyBuffs()” function from the Player script is called. This function resets all the buffs and applies them again.

It first gets a list of all the passive items currently in the inventory and calls their “ApplyEffect()” method. “ApplyEffect()” is a method from the parent class “PassiveItem” wich is overridden by every different child to apply their own custom effect.

In this example, the item “OverClocker” adds cooldown reduction to the player stats.



There are many things happening in this one-liner, that every passive item has to follow. First, it gets the Instance of the “Player” script to modify the field “coolDownReduction” from the player stats. Then it applies the effect intensity dictated by the current item level (The given index has to be the current level subtracted by one because “level” starts at 1, but the “levels” list starts at 0).

A screen shot of a computer program

Description automatically generatedAfter all the items in the inventory have applied their buffs, the function “ApplyBuffs(PlayerStats)” is called on every placed structure.

This function sets every structure attribute to be the one corresponding from the structure’s current level plus the one from the player stats (The “attackCooldown” attribute is a little special, because its value is a percentaje to be subtracted).

Once every structure has had its buffs applied, buffs not related to structures are applied, such as the player speed increase or player vision increase.

**Steps to create a new passive item that modifies an existing player stat field:**



1. Create a prefab for the item.
2. Create a new class, give it the namespace “TowerSurvivors.PassiveItems” and extend the “PassiveItem” class.

A black screen with white text

Description automatically generated

1. Override the “ApplyEffect()”, and “RemoveEffect” Methods to program behaviour.

A computer screen with white and green text

Description automatically generated

1. Once the behaviour is ready, create an Item Asset by Right click -> Create -> ScriptableObjects -> Item -> PassiveItem.

A screenshot of a computer

Description automatically generated

1. Configure the asset with the item name, description, icon, probability and reference the prefab just created.

A screenshot of a computer

Description automatically generated

1. A screenshot of a computer

   Description automatically generatedAdd levels by creating items in the array, setting each attribute. Once created the first one, copy and paste another one to make it easier and add a description for each level up consisting of which attributes changed from the previous one.
2. Go back to the structure prefab and add the reference of the item asset in the “item” field.
3. To add the item to the item pool, look for the asset named “CurrentItemPool” located in Assets -> Items.
4. Add the item asset to the Item list.

A screenshot of a computer

Description automatically generated

**Notes:**

All the attributes in the “PlayerStats” class have already been taken into account in the game’s logic. If you wanted to add an item that gives the player a buff that is not in said class, you’d have to program its behaviour, be it for structures to take it into account or modify another game behaviour. For example, to create the “Hammer” item that lets you place more structures, the “StructureManager” needs to know how many more structures can be placed, so the “ApplyBuffs()” method in the “Player” class, was modified to sum the amount from the item to the current allowed maximum structures.