3/11/2020

Bug Fixing

Project Deliverable 2

Andrew Yue-Keung Leung, Brian Ning Yu Chim, David Fernandes, Patrick Angelo Del Rosario Ocampo, Sameed Sohani
LES PROGRAMMEURS

Table of Contents

Issue Screening

Issue #6286: *PlantType.create(String) NullPointerException*

Link: https://github.com/MinecraftForge/MinecraftForge/issues/6286

Forge has an IPlantable interface that allows mod developers to define their custom plant mods to be of a specific type. There are currently 7 types that a plant could be (defined in the enum found at net.minecraftforge.common.PlantType):

- Plains → a majority of the vanilla game's plants are of this type - a plant of this type can be placed on any grass/dirt block
- Desert → a plant of this type can be placed on any sand block
- Cave → a plant of this type can be placed in cave systems on stone blocks
- Nether → a plant of this type can be placed in the Nether dimension (but not the overworld dimension)
- Beach → a plant of this type can be placed on any sand/dirt block surrounded by water
- Water → a plant of this type can be placed either on top or within water blocks
- Crop → a plant of this time can be placed on fertile dirt blocks

From the issue description provided, it seems that the reporter attempted to create a new custom plant through implementing the IPlantable interface. In their implementation of the getPlantType function, they likely had something of the following form:

```
return PlantType.create(NEW_PLANT_TYPE);
```

In Minecraft (Forge) v1.12, the PlantType.create() function used reflection (in net.minecraftforge.common.util.EnumHelper) to dynamically add values to the PlantType enum classes. As of 1.13, ASM (assembly) was preferred over reflection in the project and subsequently, the EnumHelper was replaced with the IExtensibleEnum (in net.minecraftforge.common). It appears that when porting from v1.12 to v1.13, the PlantType.create() function was not refactored properly and simply rewritten to return null. As a result, when the reporter tried to create a new PlantType, whenever a call was made to Block.canSustainPlant(), the switch statement within the function would fail due to a NullPointerException. The fact that the reporter asks if there is a workaround to create flowers which can be planted only on non-vanilla blocks is mostly irrelevant to the root problem - the issue is not with planting of flowers only on non-vanilla blocks (this is dependent on the implementations of the mod they created), the root cause is just the null returned from the PlantType. However, this case will still be accounted for as one of the acceptance tests for the sake of no ambiguity for the reporter.

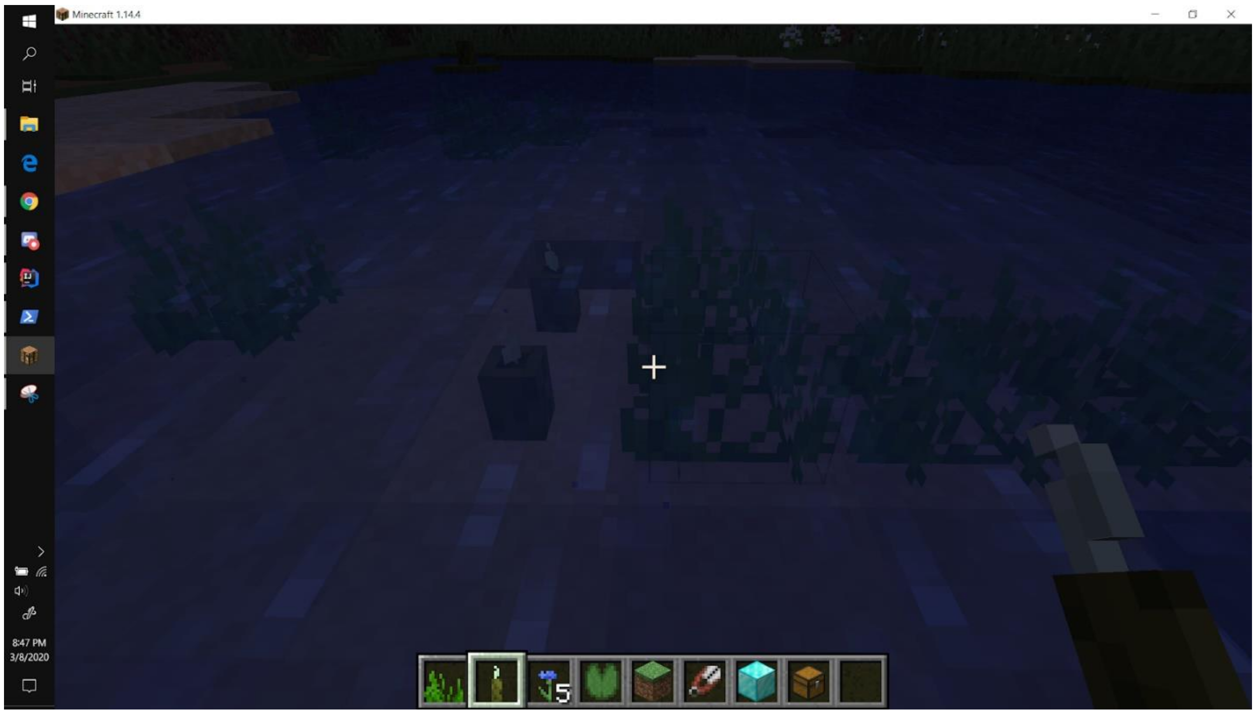
We estimate that implementing a fix for this issue would require about 2.5 hours. **More details on the breakdown of the required effort can be found in the next section**.

Issue #6325: *[1.14.4] IPlantable defaults SeaGrass and SeaPickle blocks to Plains plant type*

Link: https://github.com/MinecraftForge/MinecraftForge/issues/6325

Similar to Issue 6286, this issue also concerns the current implementation of the IPlantable interface.

Sea Pickles and Sea Grass blocks are blocks that one would generally find in bodies of water, i.e. you wouldn't find these blocks in forests or caves. At the moment, calling getPlantType on these particular blocks returns the Plains type. Having this sort of behaviour could allow players to place Sea Grass and Sea Pickles on top of blocks that aren't underwater, which doesn't make logical sense (unless one wanted this sort of behaviour). Below is a picture of ideal block placement behaviour of a Sea Pickle block (to the left of the crosshair) and the Sea Grass block (to the right of the crosshair).



We estimate that implementing a fix for this issue would require about 90 minutes of work overall. **More details on the breakdown of the required effort can be found in the next section**.

Issue #6326: *[1.14.4] Remove misleading @Nullable from IForgeBlock::canCreatureSpawn*

Link: https://github.com/MinecraftForge/MinecraftForge/issues/6326

Forge has an IForgeBlock interface that aids mod developers with the creation of their own custom blocks. It contains a method (canCreatureSpawn) that returns a boolean of whether or not a creature can spawn on a given block. One of its parameters - entityType, of type EntityType<?>, is annotated with a @Nullable, and this interface directly calls canEntitySpawn with entityType as one of its parameters. However, in the vanilla Minecraft's MagmaBlock implementation, this annotation is missing. A possible consequence of this would be that if the game was trying to determine whether a given entity could spawn on a magma block and this entity turned out to be null, a NullPointerException would occur and the game would crash.

We estimate that implementing and verifying this change would take at most 70 minutes. **More details on the breakdown of the required effort can be found in the next section**.

Issue #6344: *Biome ID over 255, voroni zoom layer*

Link: https://github.com/MinecraftForge/MinecraftForge/issues/6344

In Minecraft, mod developers have the option for creating their own custom biomes (large areas of land with its own climate and vegetation). At the moment, if one were to install more than 256 custom mod biomes into the game, the world would not generate properly. This particular issue comes from the fact that there is a limit to how many biomes can be present in the game - the setting of this limit can be found in the VoroniZoomLayer enum class (responsible for helping with the generation of the Minecraft world). The issuer has made a recommended fix for the issue (removing the bitwise operation on an integer at a specific line), but is still testing it to see if it brings about any other issues.

We estimate that implementing and verifying this change would take about 3 hours. This includes:

- Making the suggested fix in the code (< 10 minutes)

- Creating the test cases and testing the code thoroughly to make sure nothing else broke with world generating (~ 2 hours and 50 minutes)

Issue #6405: *Registering villager types per biome is impossible*

Link: https://github.com/MinecraftForge/MinecraftForge/issues/6405

Villagers in Minecraft are instantiated with trade tables, each storing price, item and availability. Each trade type implements *VillagerTrades.ITrade* allowing for multiple different types of storage. One specific villager, the Fisherman uses the class *EmeraldForVillageTypeItemTrade* which stores a map based on VillageType, i.e.:

```
Map<IVillagerType, Item>.builder()
 .put(IVillagerType.PLAINS, Items.OAK_BOAT)
 ...
 .put(IVillagerType.SWAMP, Items.DARK_OAK_BOAT)
 .build();
```

This map is passed into the constructor of the class which checks if every registered biome in the game has a valid item to trade. In the base game of Minecraft, hardcoding these VillagerTypes → Item pairs is acceptable, given the limited biomes in the game. However, modders are able to add new biomes to the game - causing the initial check to fail, given that one cannot access this constructor nor add to the map prior to the constructor running; an exception is thrown causing the game to crash.

We estimate that implementing a fix for this particular issue would take about 6-7 hours. This includes:

- Creating a quick custom mod biome with a custom villager type in order to replicate the issue (~ 1 hour)
- Integrating with the current standard MinecraftForge EventBus for villager Trades (~ 2 - 3 hours)
- Editing the Minecraft class as minimally as possible to stop the crash yet preserve normal code. (~ 2 hours)
- Creating test cases and thoroughly testing the code to make sure we do not have the game crash/introduce new bugs into the game (~ 1 hour)

Issue #6040: *[1.14.4] Custom banner patterns no longer work, additional patches needed*

Link: https://github.com/MinecraftForge/MinecraftForge/issues/6040

When Mojang released version 1.14 of Minecraft, they implemented a new system for creating custom banner patterns. The Loom is essentially a block that allows players to input 3 items - a blank banner object, a color dye object, and a custom pattern stencil object - and craft a banner with a custom design out of said items. Before this update, the method for crafting custom banner patterns involved bringing the materials together into a crafting table. This method did not involve the use of any stencil object, so if a mod developer wanted to implement their own custom pattern, they would do so by specifying a custom crafting table recipe that yielded some modded pattern.

The introduction of the Loom block was more or less an overhaul to the old system, and this created problems for any pre-1.14 custom pattern mod. If a player tried to make a custom pattern stencil that did not already exist in the game with specific items, the game would crash. It has been noted on the GitHub issue that this comes as a result of Mojang making the decision to hard code certain values, which makes it a bit more complicated for a mod developer to make their own custom pattern.

We estimate that implementing a fix for this issue would require about a full day's worth of work (we'd be taking into account the group's strengths/weaknesses to estimate the time). This includes:

- Going through each component involved with crafting a custom banner pattern and understanding how it works (~ 1 hour)
- Designing a new system that allows for modders to interface with the custom banner patterns in the game (~ 12 - 16 hours)
    - This would also entail figuring out how to "hack" into Minecraft's vanilla code and work around the hard-coded values that have been set
- Implementing the design (~ 4 hours)
- Writing test cases, testing our new implementations in the game, and making any changes (~ 3 - 7 hours)

Issue Selection

Issue #6286: *PlantType.create(String) NullPointerException*

Link: https://github.com/MinecraftForge/MinecraftForge/issues/6286

We chose this issue as one of the bugs to fix based on the scale at which it affects the codebase and its importance to the game. Planting in Minecraft is a vital component for players that want to create farms, and more importantly, mod developers that want to create custom plant types. Having this issue unaddressed prevents mod developers from doing so, making players settle for the game's vanilla plants. The amount of components that are relevant to this issue is of a decent size. This in turn makes it an easy task for us first-time FOSS project contributors to sift through the code and gain a better understanding of the code related to planting in Minecraft. As a result, we expect this to allow us to pinpoint the issue and create an appropriate solution.

The risks associated with this issue are relatively low in the sense that mod developers for custom plants wouldn't have to touch too much code pertaining to plant types. Given the fact that a mod developer only needs to reimplement the getPlantType and canSustainPlant methods in the IPlantable interface and Block class respectively, we do not have to worry too much about components breaking in the overall system. We just need to be sure that we refactor the PlantType.create() method in such a way that allows for mod developers to easily create custom plant types. All in all, the changes that we make along with the required debugging efforts should be suitable tasks that fit our timeframe for this deliverable.

As mentioned in the previous section, we expected the fix for this issue to be roughly 2.5 hours in total. This would include:

- Investigating the previous implementation and the current implementation for adding values to Enums (~ 1 hour)
- Creating a test mod which includes a custom plant block and a custom "soil" block (block which can have the plant planted on) and testing that the implementation was as expected (~ 1.5 hours)

Issue #6325: *[1.14.4] IPlantable defaults SeaGrass and SeaPickle blocks to Plains plant type*

Link: https://github.com/MinecraftForge/MinecraftForge/issues/6325

Similar to the reason given for issue #6326 (the "misleading" @Nullable annotation), we chose this issue for its difficulty towards newbie FOSS project contributors. Given that this is our first time contributing to an open source project, we want to familiarize ourselves with the codebase by getting our feet wet with small scale issues. Given the large codebase and lack of proper documentation, sifting through the components and trying to gain a better understanding of everything adds an extra layer of work for us to tackle. One would think the work conducted in D1 would provide aid for D2, but simply going through an architectural analysis is not enough - making changes to the code, building the project several times over, writing tests, and examining the behaviour are also necessary for gaining a more holistic understanding of the project.

One risk that comes with addressing this issue is the potential to change the behaviour of how Sea Grass and Sea Pickles work in the game. There is another plant block with the water type - the Lily Pad. This is a block that is generally placed on top of a water block (like an actual lily pad in real life). Giving Sea Grass and Sea Pickles the water type could result in players being able to place them on top of water, making it look as if they're "floating." Apart from how they interact with water blocks, there is also a chance that it could change how blocks submerged in water deal with Sea Grass/Pickles being planted on top of them. These issues could essentially lead into us creating a plethora of other problems that need to be addressed, consuming more time and impeding our progress. It is for these reasons that we need to be wary of how our changes will affect other components in the project.

From the description provided in the previous section, we made an estimate that it would take about 90 minutes to implement a fix for this issue. This would include:

- Going through the components affected by the issue and understanding how everything uses the IPlantable interface (~ 30 minutes)
- Writing out the necessary code that fixes the issue (~ 20 minutes)
- Writing the test mods and executing them in-game (~ 40 minutes)
    - The time it takes to build and run the Minecraft/Forge client varies by the specs of each developer's computer

All in all, it would not require too much of a strenuous effort to address this issue, making it an ideal pick for this deliverable. As with issue 6326, this is one of our stepping stones towards gaining a better understanding of the project and familiarizing ourselves with the contribution process.

Issue #6326: *[1.14.4] Remove misleading @Nullable from IForgeBlock::canCreatureSpawn*

Link: https://github.com/MinecraftForge/MinecraftForge/issues/6326

The main reason this issue was selected was to familiarize ourselves with the process of contributing to MinecraftForge in general since it is not as simple as forking, cloning, branching, committing, and submitting a pull request. Setting up the environment correctly is complicated and it does not help that the documentation is out of date. Furthermore, we do not commit changes to Minecraft's vanilla code. Instead, we have to generate and commit patch files.

The only risk we see with working on this issue is that removing the @Nullable annotation could potentially cause other parts of the code that we are not yet familiar with to no longer work; which is why instead of following the suggested fix, we will implement our own safer fix.

Overall, we estimate this issue to not require an extensive amount of effort since as beginning contributors to the project, the choice of working on this issue is solely to gain learning experience while we continuously learn more about the architecture of MinecraftForge. For a more in-depth breakdown of the required work (70 minutes), we have the following:

- Going through the relevant components and understanding how they use the IForgeBlock interface (~ 30 minutes)
- Writing out the necessary code that fixes the issue (~ 10 minutes)
- Writing the test mods and executing them in the game (~ 30 minutes)

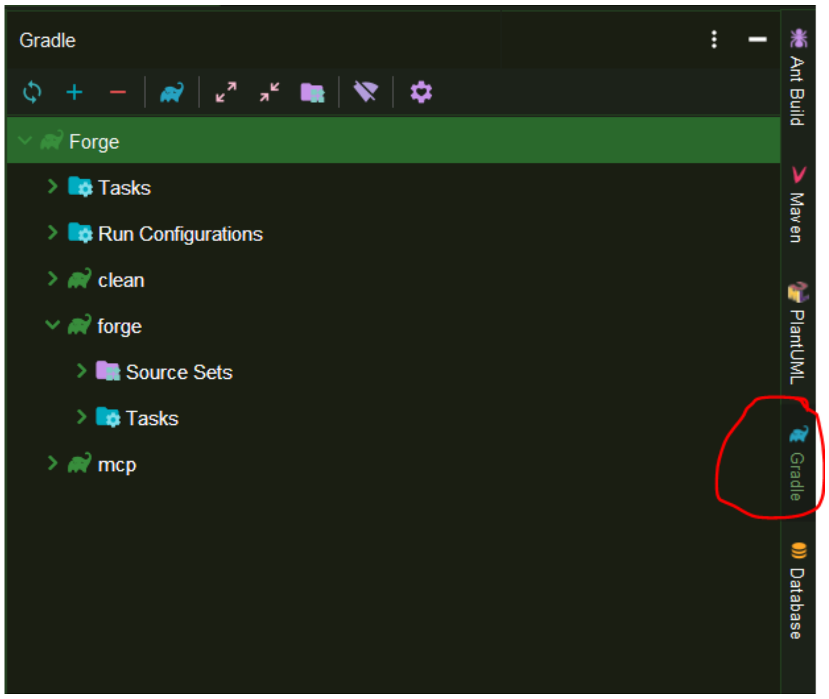
In addition to the estimates we've made for the required work on each issue, there is an extra hour or two required for learning how to create test cases in Minecraft Forge via test mods. More on this will be described later under the Documentation section.

Documentation

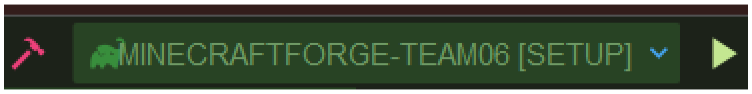
Setting up your Environment

NOTE - we assume that you are using IntelliJ for this setup, as it is the superior IDE when developing in Java. Also, we are compiling with Java 1.8.

1. Clone the forked repository
   a. git clone https://github.com/CSCD01/MinecraftForge-team06.git
2. Check out the 1.14 branch
   a. git checkout 1.14.x
3. Open the project on IntelliJ as a gradle project. Leave all the default inputs as they are in the Wizard.
4. Allow for the gradle project to run the initial build. After it has finished indexing the files, click the Gradle project tab on the side. You should obtain the following:



5. Under Tasks/other, right click the task “setup”
    a. Enter “clean setup” for the tasks textbox
    b. Enter “-Xmx3G -Xms3G” for the VM Options textbox
    c. Click Apply and OK
6. At the top, make sure the combo box is set to the configuration you’ve just made and press the green triangle.

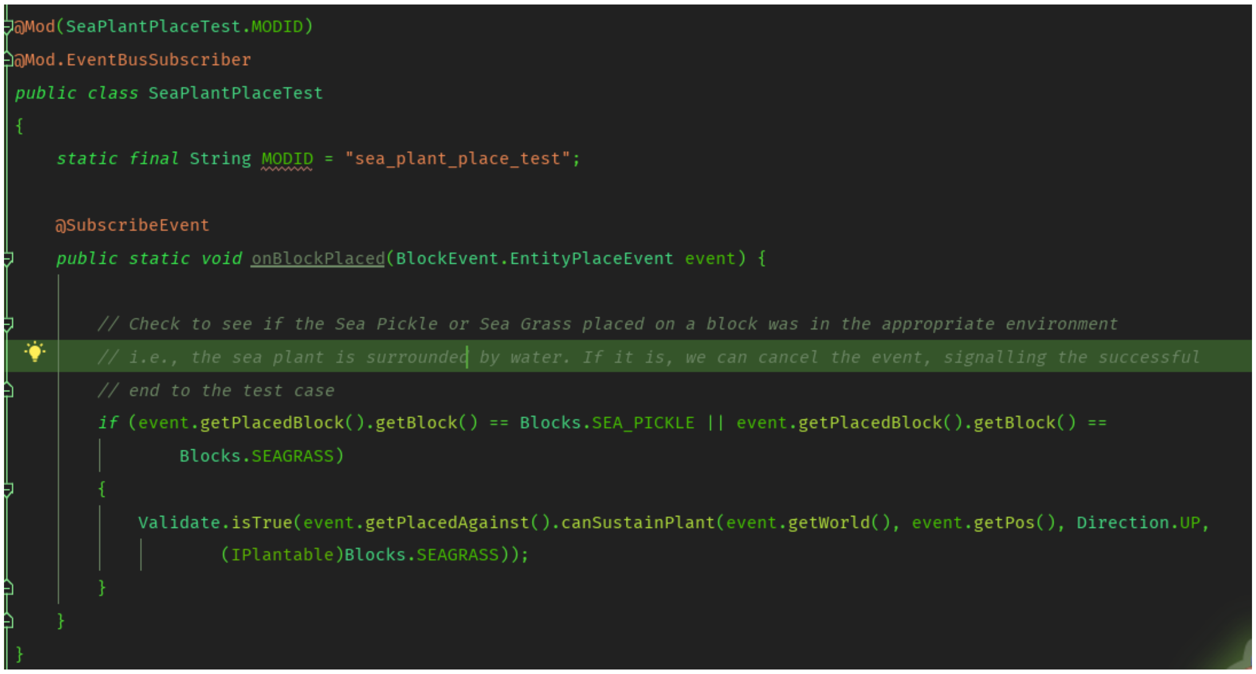


7. Let the project run the build. Once it has finished, click File → Project Structure → Modules and ensure that under Forge/forge, you have the following modules: fmllauncher, main, test, and userdev. If they are, then press OK.
8. Click the combo box that you clicked on before (between the hammer and green triangle) and click “Edit Configurations.”
9. Click “Application” and for the apps “clean clean client” and “clean clean server”, ensure that the “Use classpath of module” is set to “Forge.clean.main”. If this yields issues, use “Forge.forge.main” instead. Click Apply for both apps and press OK.
10. Select “forge forge client” from the combo box and press the green button. It should take 5-10 minutes for the Minecraft client to launch.

Enabling the Test Mods (a.k.a. the Test Cases)

A mod in Minecraft is generally a user-made modification that either adds more content or changes the overall behaviour of the game. It is for this particular reason that we can use mods as a means to test the changes we make to Forge. As mentioned before, test mods are essentially the test cases in Forge. If we make a change and want to verify its behaviour, we can create a mod to that helps us in doing so (for example - creating a mod that makes a player take damage whenever a block was placed in order to check for a specific condition). If a contributor were to make changes to any of Forge’s codebase, they would be required to write a test mod in order to make sure their changes did not introduce any new issues to the game and/or Forge itself.

Each test mod is a single java class that generally has the following structure:



```
@Mod(SeaPlantPlaceTest.MODID)
@Mod.EventBusSubscriber
public class SeaPlantPlaceTest
{
    static final String MODID = "sea_plant_place_test";

    @SubscribeEvent
    public static void onBlockPlaced(BlockEvent.EntityPlaceEvent event) {

        // Check to see if the Sea Pickle or Sea Grass placed on a block was in the appropriate environment
        // i.e., the sea plant is surrounded by water. If it is, we can cancel the event, signalling the successful
        // end to the test case
        if (event.getPlacedBlock().getBlock() == Blocks.SEA_PICKLE || event.getPlacedBlock().getBlock() ==
                Blocks.SEAGRASS)
        {
            Validate.isTrue(event.getPlacedAgainst().canSustainPlant(event.getWorld(), event.getPos(), Direction.UP,
                    (IPlantable)Blocks.SEAGRASS));
        }
    }
}
```

Test mods consist of test methods that are subscribed to the Event Bus that notifies its subscribers when a certain event has occurred. When a test method has been "notified", they will be triggered and will proceed to execute the code within them. In these methods, one can collect data related to the event. Taking the above test as an example, the EntityPlaceEvent can tell you what kind of block has been placed, if it can sustain specific plants, the world in which it has been placed in, and so forth. As per the name, the event will be triggered whenever an entity (player or AI) places an object/block on another block in the game. From the data that has been collected, one can then proceed to run validations and checks to see that everything is returning the correct values and behaving appropriately.

To "run" these test cases, you will need to load the test mod into the game and perform actions that will trigger the test cases to execute and perform the checks/validations.

Please refer to the "Enabling test mods section" of this document →
https://mcforge.readthedocs.io/en/1.14.x/forgedev/

The following video is also a good resource for setting this up →
https://www.youtube.com/watch?v=pLWQk6ed56Q

Bug Fixes

Issue #6286: *PlantType.create(String) NullPointerException*

Branch: 1.14.x_issue-6286_fix
Link: https://github.com/CSCD01/MinecraftForge-team06/tree/1.14.x_issue-6286_fix

PlantType.java (generated after environment has been setup and code is decompiled)

- PlantType now implements IExtensibleEnum, allowing for custom plant types to be dynamically added into the game via mods.
- PlantType.create() has been refactored to throw an exception if the enum has not been extended (as opposed to returning null and yielding the Null Pointer Exceptions from before). On inspecting how other Enums implemented dynamic creation of values, we came across IExtensibleEnum. IExtensibleEnum is an interface which allows for the Forge code to use ASM to dynamically add values into Enums. On further investigation of the docstring of IExtensibleEnum as well as how other classes implemented dynamic Enums (for example net.minecraft.enchantment.EnchantmentType, net.minecraft.enchantment.EntityClassification, net.minecraft.enchantment.Rarity), each of these Enums also implement IExtensibleEnum in the same way (by throwing an IllegalStateException which is then caught and replaced by ASM at runtime to create new values).

Test Cases

- Refer to **/src/test/java/net/minecraftforge/debug/block/CustomPlantTypeTest.java** for more details
- The CustomPlantTypeTest is a test mod that registers a custom block and a custom plant of a custom type into the game. The goal of this test mod is to check and see if both a vanilla block AND a custom block can support a custom plant type. If the game does not crash (i.e., no null pointer exceptions arise) when a player tries to place the custom plant onto either a vanilla or custom block, then the test passes. Note: the "custom plant" will look exactly like the custom block but will function as a flower.
- To run this test, you must:
    - Load the mod into Minecraft (change run configuration to forge forge test client)
    - Create a new world (or enter an existing one) in Creative Mode

    - Type in the command
      /give @a custom_plant_type_test:test_custom_block
      /give @a custom_plant_type_test:test_custom_plant
    - Place the custom block anywhere on flat land
    - Make an attempt to place the custom plant on the custom block and any other block (grass, dirt, stone, etc.) in the game
        - If attempting to place on the custom block, it should plant the custom plant correctly. (Remember that it will look like the custom block however)
        - If attempting to place on anything else, the custom plant should not be planted.
        - If the game doesn't crash in either situation, the test has passed.

Issue #6325: *[1.14.4] IPlantable defaults SeaGrass and SeaPickle blocks to Plains plant type*

Branch: 1.14.x_issue-6325_fix
Link: https://github.com/CSCD01/MinecraftForge-team06/tree/1.14.x_issue-6325_fix

IPlantable.java

- Extra checks have been added for the SEAGRASS and SEA_PICKLE blocks on lines 43 and 44 so that we can return PlantType.Water for both blocks.

Test Cases

- Refer to **/src/test/java/net/minecraftforge/debug/block/SeaPlantPlaceTest.java** for more details
- The SeaPlantPlaceTest is a test mod that uses the EventBus and has a test method (onBlockPlaced) that acts as a listener for the event in which an entity places a block in the game. When an entity places a block, this method gets triggered first checks to see if the block placed was Sea Grass or a Sea Pickle. It then checks to see if the block in which those sea plants are sitting can sustain them.
- To run this test, you must:
    - Load the mod into Minecraft (forge forge test client)
    - Enter Creative Mode

        - Select the Sea Grass or Sea Pickle block from the creative and place them anywhere in your quick inventory
        - Look for a body of water
        - Make an attempt to place either of the sea plants on the floor of the body of water
            - If the game does not crash, then the test has passed

Issue #6326: *[1.14.4] Remove misleading @Nullable from IForgeBlock::canCreatureSpawn*

Branch: 1.14.x_issue-6326_fix
Link: https://github.com/CSCD01/MinecraftForge-team06/tree/1.14.x_issue-6326_fix

MagmaBlock.java (generated after environment has been setup and code is decompiled)

- On line 75, a null check has been added for the EntityType object so that we do not yield NullPointerExceptions.

Test Cases

- Refer to **/src/test/java/net/minecraftforge/debug/block/MagmaBlockSpawnTest.java**
- Similar to the SeaPlantPlaceTest, we have a test mod that has a test method (onEntitySpawned) acting as a listener for the event in which an entity spawns into the world. When an entity (such as a zombie, skeleton, or creeper) spawns into the world, this method makes a call to MagmaBlock's canEntitySpawn method with a value of "null" for the entity type and checks to see that it returns false.
- There is another test method (onZombieSpawned) that has similar behaviour, but passes in "ZOMBIE" for the canEntitySpawn method under the entity type field and checks to see that it returns true.
- To run this test, you must:
    - Load the mod into Minecraft (forge forge test client)
        - onEntitySpawned
            - Wait for any entity to spawn
            - If the game does not crash, it means that the method successfully performed the null check with an entity type of null

- onZombieSpawned
    - Wait for a zombie to spawn
    - If the game does not crash, it means that canEntitySpawn returned the correct boolean value

In order to run our tests, you will need to checkout and build the code from the appropriate branch. For example - to run the PlantType tests (issue 6286), you will need to checkout the branch 1.14.x_issue-6286_fix.

Technical Commentary

Issue #6286: *PlantType.create(String) NullPointerException*

Link: https://github.com/MinecraftForge/MinecraftForge/issues/6286

Even though this fix was more or less a one line solution, we have now made the PlantType enum extensible. As it was mentioned before, this change grants mod developers the ability to add their own custom plant types into the game without having to worry about any crashes that come as a result of null pointer exceptions. Mod developers can now easily interact with the vanilla game's plant system. They can do this by reimplementing a few methods that interact with PlantType (namely canSustainPlant and getPlantType). Overall, we've refactored the code in such a way that we don't have to worry about mod developers encountering null pointer exceptions in this particular area.

Source code files added/modified/removed:

Added: /src/test/java/net/minecraftforge/debug/block/CustomPlantTypeTest.java

Modified: /src/main/java/net/minecraftforge/common/PlantType.java

Modified: /src/test/resources/META-INF/mods.toml

Issue #6325: *[1.14.4] IPlantable defaults SeaGrass and SeaPickle blocks to Plains plant type*

Link: https://github.com/MinecraftForge/MinecraftForge/issues/6325

Apart from adding a couple more lines to the codebase and a few extra test mod files, our changes essentially add another set of cases/conditions to check. In regards to the IPlantable interface, we add an extra check for Sea Grass and Sea Pickles so that they can be recognized as plants of the Water type. In addition to adding that extra check in the interface, we also had to slightly modify the behaviour of the method that determines whether or not a block can sustain specific plants. This is to ensure that the mentioned sea plant objects are dealt with accordingly when trying to plant them on certain blocks. Overall, this ensures that those particular sea plants are growing in the appropriate environment within the game, i.e. underwater and not on dry land/anywhere else.

Source code files added/modified/removed:

Added: /src/test/java/net/minecraftforge/debug/block/SeaPlantPlaceTest.java

Modified: /src/main/java/net/minecraftforge/common/IPlantable.java

Modified: /src/test/resources/META-INF/mods.toml

Issue #6326: *[1.14.4] Remove misleading @Nullable from IForgeBlock::canCreatureSpawn*

Link: https://github.com/MinecraftForge/MinecraftForge/issues/6326

As mentioned previously, we felt that there was another fix that is safer than simply removing the @Nullable annotation. Instead, we added a null check to the EntityType parameter in the canEntitySpawn method of the vanilla Minecraft MagmaBlock prior to calling EntityType.isImmuneToFire(). This avoids the NullPointerException problem mentioned earlier and potential crashes/unwanted behaviour in the game.

Source code files added/modified/removed:

Added: /src/test/java/net/minecraftforge/debug/block/MagmaBlockSpawnTest.java

Added: /patches/minecraft/net/minecraft/block/MagmaBlock.java.patch

Modified: /src/test/resources/META-INF/mods.toml

Software Development Process

As it was mentioned in our previous deliverable, we are following Kanban for our software development process. On our board, we have 5 columns - “To Do”, “Blocked”, “In Progress (5)”, “Needs Verification (5)”, “Under Verification (3)”, and “Done.”

- To Do → all the items that we need to complete during Deliverable n, where n (- R
- Blocked → all the items that cannot be worked on due to some external factor (for example, the developers for Minecraft Forge introduced a bug, inhibiting contributors from working on a stable build and addressing existing issues)
- In Progress (5) → the items that are currently being worked on
    - We chose a value of 5 for the limit of how many tasks can be in this column
        - There are 5 members in the team, and each member will be working on a specific task
- Needs Verification (5) → the items that have been completed, but need to be reviewed by at least one other team member
    - We chose a value of 5 for the limit of how many tasks “need verification” (same reasoning from the In Progress column applies here
- Under Verification (3) → the items that have been completed and are currently being reviewed
    - We chose a value of 3 for this column because we want to ensure we are paying close attention to the detail of the work that has been completed in each task in order to ensure that we are delivering quality content.
- Done → the items that have been verified

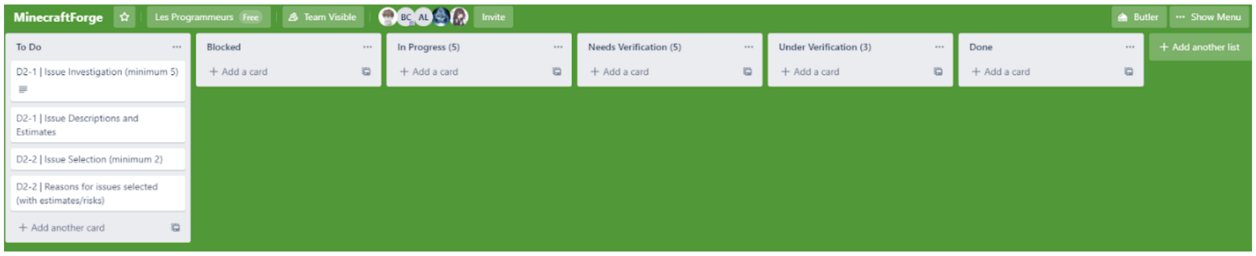
At the beginning of this deliverable, we began by collectively looking at all of the issues in the project’s backlog and started picking out ones that would fit our team’s capabilities and schedule. After selecting those issues, we began conducting further research into them in order to determine what exactly it was that needed to be done. A card was made for each issue on our Trello board, and a checklist was established for each issue as a means to track the progress on those tasks. This allowed us to get a better visualization of tasks that needed to be completed along with how much progress we’ve made in the deliverable.

Kanban helped our group become more efficient as it helped us visualize where the work was and how it was progressing (without the actual need to hold a face-to-face meeting daily). As a result of this, it helped us save us time and unnecessary updates while progress was still consistently being made. One particular benefit to making use of the Kanban board for example, was that it helped our team progress faster. Rather than an individual finishing their task and waiting for a team update to decide on what the next task to undertake was, an individual could directly go to the board and view where the team needed more hands.

Aside from efficiency, one other benefit that occurred was that the team dynamic became less about assigning work to individuals to work independently and instead became more-so about pulling work to do and other team members offering to help when they see a visible blockage. This allowed the team to become closer and feel that each ticket belonged to the team rather than just an individual. As we work into the future, for example, in adding a feature, we feel this will benefit us.

Below are screenshots of phases that our board underwent during the whole process:

1) Beginning of Deliverable 2 → Initial Board State



MinecraftForge | Les Programmeurs Free | Team Visible | BC AL | Invite | Butler | Show Menu

To Do
- D2-1 | Issue Investigation (minimum 5)
- D2-1 | Issue Descriptions and Estimates
- D2-2 | Issue Selection (minimum 2)
- D2-2 | Reasons for issues selected (with estimates/risks)
- + Add another card

Blocked
- + Add a card

In Progress (5)
- + Add a card

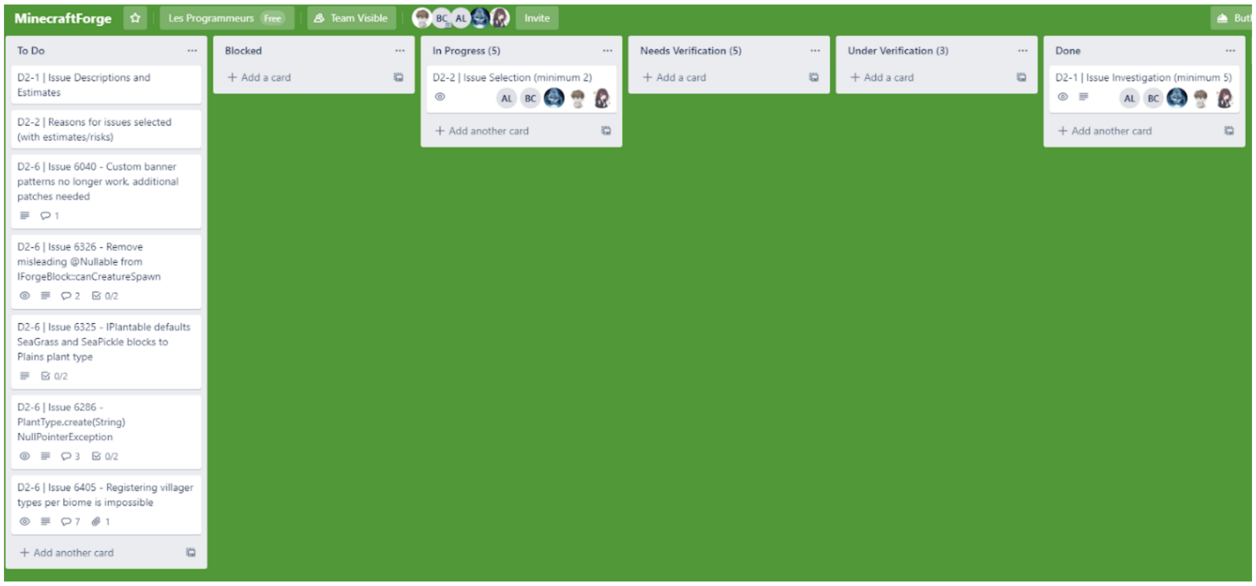
Needs Verification (5)
- + Add a card

Under Verification (3)
- + Add a card

Done
- + Add a card

+ Add another list

2) Picking out promising issues and figuring out which ones we wanted to work on (and also establishing subtasks for the tasks that were shortlisted)



MinecraftForge | Les Programmeurs Free | Team Visible | BC AL | Invite | Butl

To Do
- D2-1 | Issue Descriptions and Estimates
- D2-2 | Reasons for issues selected (with estimates/risks)
- D2-6 | Issue 6040 - Custom banner patterns no longer work, additional patches needed — 1
- D2-6 | Issue 6326 - Remove misleading @Nullable from IForgeBlock::canCreatureSpawn — 2, 0/2
- D2-6 | Issue 6325 - IPlantable defaults SeaGrass and SeaPickle blocks to Plains plant type — 0/2
- D2-6 | Issue 6286 - PlantType.create(String) NullPointerException — 3, 0/2
- D2-6 | Issue 6405 - Registering villager types per biome is impossible — 7, 1
- + Add another card

Blocked
- + Add a card

In Progress (5)
- D2-2 | Issue Selection (minimum 2) — AL BC
- + Add another card

Needs Verification (5)
- + Add a card

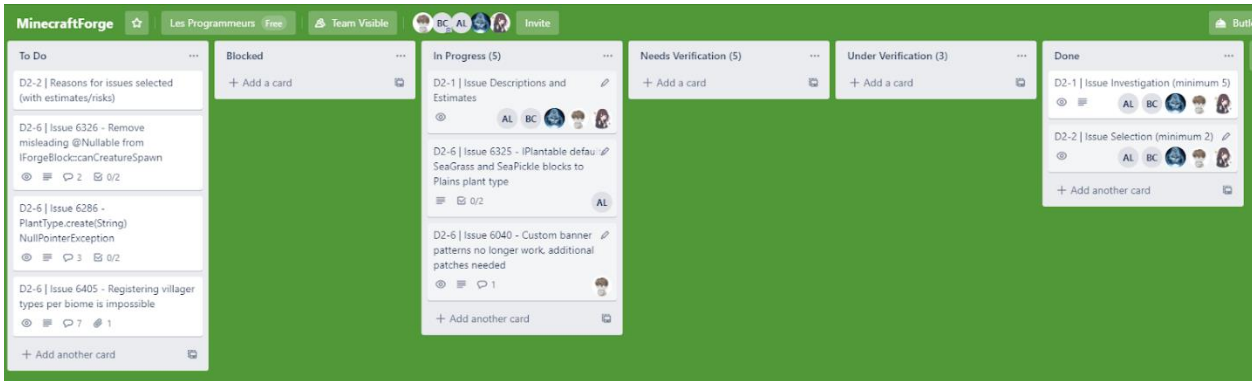
Under Verification (3)
- + Add a card

Done
- D2-1 | Issue Investigation (minimum 5) — AL BC
- + Add another card

For the issues that we chose (6286, 6325, 6326), there were 2 main subtasks that needed to be completed (implementing/documentation and creating the appropriate acceptance tests)

3) Progress begins for Issue 6325 and 6040

(see next page)



MinecraftForge | Les Programmeurs Free | Team Visible | Invite

To Do
- D2-2 | Reasons for issues selected (with estimates/risks)
- D2-6 | Issue 6326 - Remove misleading @Nullable from IForgeBlock::canCreatureSpawn — 2 — 0/2
- D2-6 | Issue 6286 - PlantType.create(String) NullPointerException — 3 — 0/2
- D2-6 | Issue 6405 - Registering villager types per biome is impossible — 7 — 1
- + Add another card

Blocked
- + Add a card

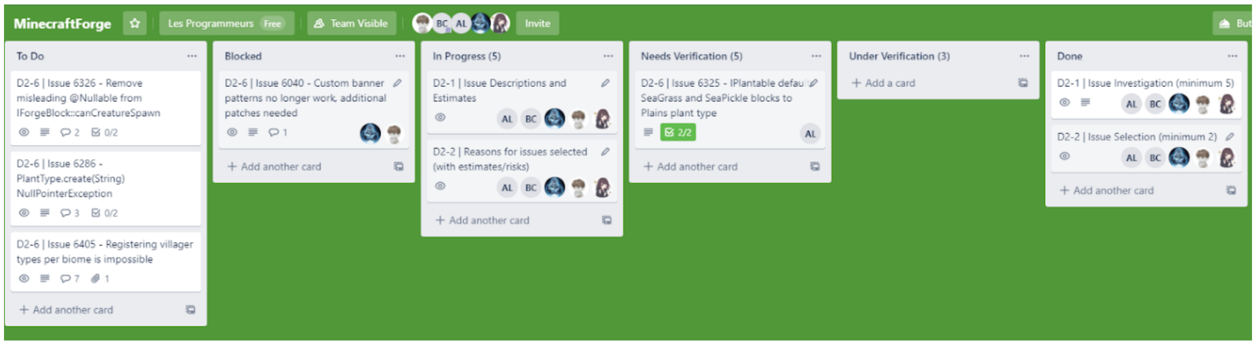
In Progress (5)
- D2-1 | Issue Descriptions and Estimates — AL BC
- D2-6 | Issue 6325 - IPlantable defau SeaGrass and SeaPickle blocks to Plains plant type — 0/2 — AL
- D2-6 | Issue 6040 - Custom banner patterns no longer work, additional patches needed — 1
- + Add another card

Needs Verification (5)
- + Add a card

Under Verification (3)
- + Add a card

Done
- D2-1 | Issue Investigation (minimum 5) — AL BC
- D2-2 | Issue Selection (minimum 2) — AL BC
- + Add another card

4) Issue 6325 is completed; issue 6040 is moved to blocked after conducting further research on the issue and seeing that there was a lot more work that needed to be done



MinecraftForge | Les Programmeurs Free | Team Visible | Invite

To Do
- D2-6 | Issue 6326 - Remove misleading @Nullable from IForgeBlock::canCreatureSpawn — 2 — 0/2
- D2-6 | Issue 6286 - PlantType.create(String) NullPointerException — 3 — 0/2
- D2-6 | Issue 6405 - Registering villager types per biome is impossible — 7 — 1
- + Add another card

Blocked
- D2-6 | Issue 6040 - Custom banner patterns no longer work, additional patches needed — 1
- + Add another card

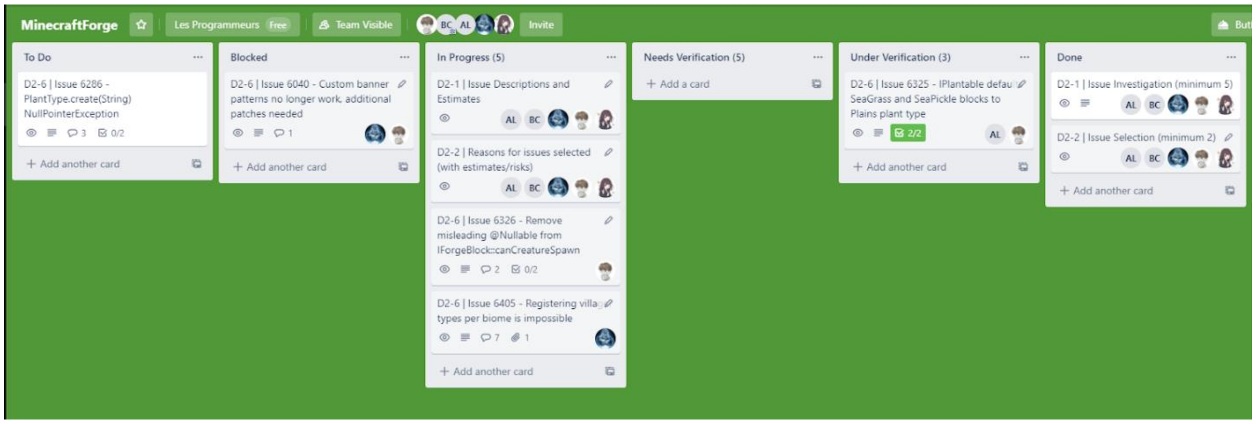
In Progress (5)
- D2-1 | Issue Descriptions and Estimates — AL BC
- D2-2 | Reasons for issues selected (with estimates/risks) — AL BC
- + Add another card

Needs Verification (5)
- D2-6 | Issue 6325 - IPlantable defau SeaGrass and SeaPickle blocks to Plains plant type — 2/2 — AL
- + Add another card

Under Verification (3)
- + Add a card

Done
- D2-1 | Issue Investigation (minimum 5) — AL BC
- D2-2 | Issue Selection (minimum 2) — AL BC
- + Add another card

5) Issue 6325 is under verification, work begins on issue 6326 and issue 6405



MinecraftForge | Les Programmeurs Free | Team Visible | Invite

To Do
- D2-6 | Issue 6286 - PlantType.create(String) NullPointerException — 3 — 0/2
- + Add another card

Blocked
- D2-6 | Issue 6040 - Custom banner patterns no longer work, additional patches needed — 1
- + Add another card

In Progress (5)
- D2-1 | Issue Descriptions and Estimates — AL BC
- D2-2 | Reasons for issues selected (with estimates/risks) — AL BC
- D2-6 | Issue 6326 - Remove misleading @Nullable from IForgeBlock::canCreatureSpawn — 2 — 0/2
- D2-6 | Issue 6405 - Registering villa types per biome is impossible — 7 — 1
- + Add another card

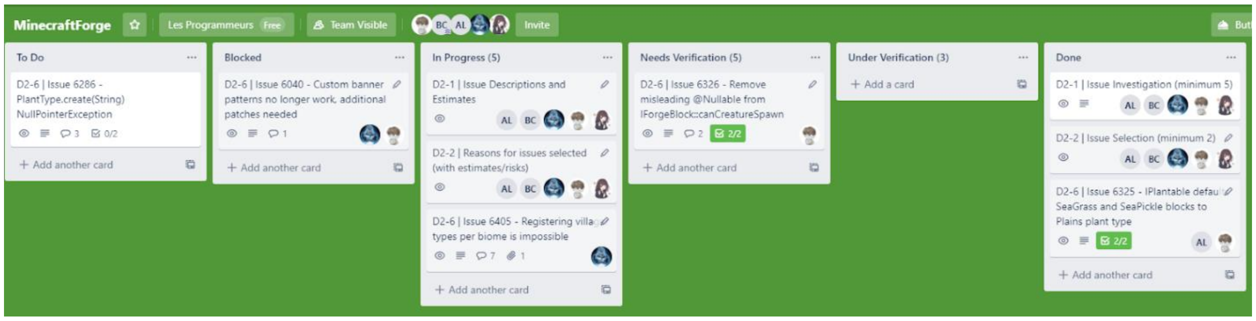
Needs Verification (5)
- + Add a card

Under Verification (3)
- D2-6 | Issue 6325 - IPlantable defau SeaGrass and SeaPickle blocks to Plains plant type — 2/2 — AL
- + Add another card

Done
- D2-1 | Issue Investigation (minimum 5) — AL BC
- D2-2 | Issue Selection (minimum 2) — AL BC
- + Add another card

6) Issue 6326 is completed and requires verification; issue 6325 is verified and moved to done

(see next page)


MinecraftForge | Les Programmeurs Free | Team Visible | Invite

To Do
- D2-6 | Issue 6286 - PlantType.create(String) NullPointerException — 3, 0/2

Blocked
- D2-6 | Issue 6040 - Custom banner patterns no longer work, additional patches needed — 1

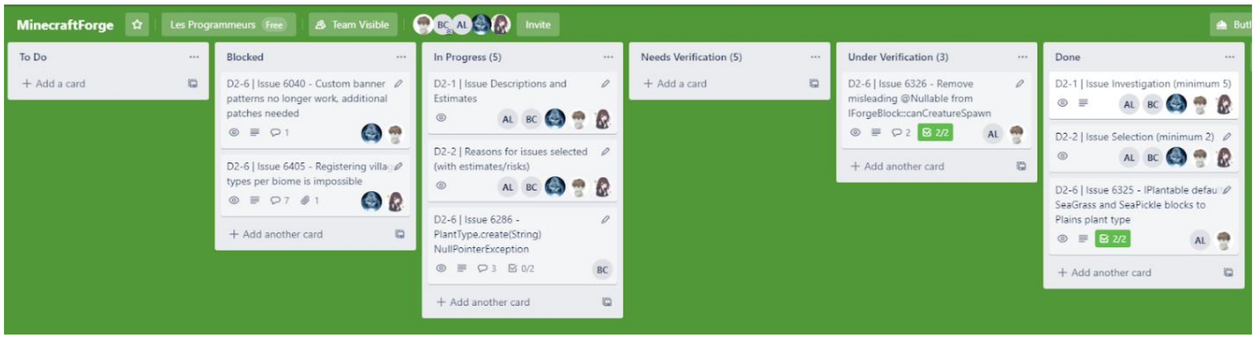
In Progress (5)
- D2-1 | Issue Descriptions and Estimates
- D2-2 | Reasons for issues selected (with estimates/risks)
- D2-6 | Issue 6405 - Registering villa types per biome is impossible — 7, 1

Needs Verification (5)
- D2-6 | Issue 6326 - Remove misleading @Nullable from IForgeBlock::canCreatureSpawn — 2, 2/2

Under Verification (3)
- + Add a card

Done
- D2-1 | Issue Investigation (minimum 5)
- D2-2 | Issue Selection (minimum 2)
- D2-6 | Issue 6325 - IPlantable defau SeaGrass and SeaPickle blocks to Plains plant type — 2/2

7) Issue 6326 is under verification; issue 6405 is moved to blocked for the same reason as issue 6040


MinecraftForge | Les Programmeurs Free | Team Visible | Invite

To Do
- + Add a card

Blocked
- D2-6 | Issue 6040 - Custom banner patterns no longer work, additional patches needed — 1
- D2-6 | Issue 6405 - Registering villa types per biome is impossible — 7, 1

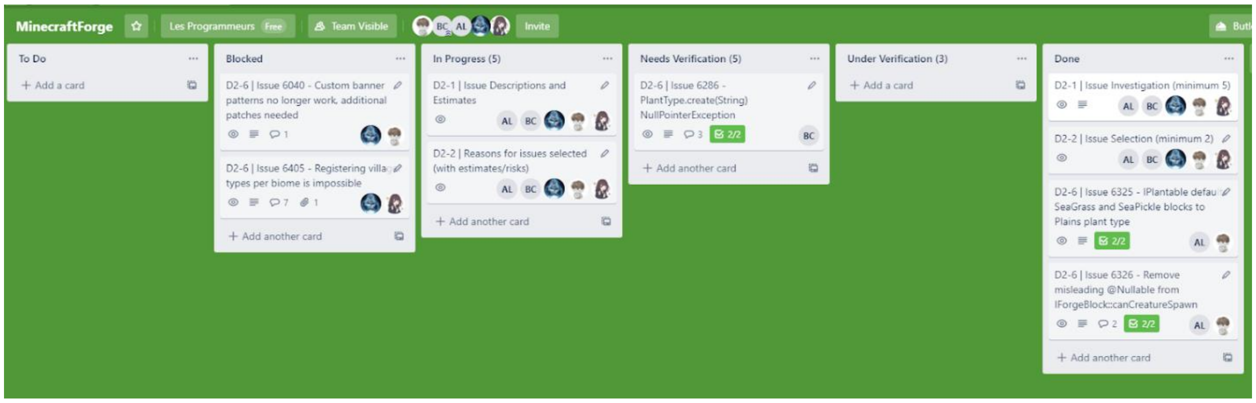
In Progress (5)
- D2-1 | Issue Descriptions and Estimates
- D2-2 | Reasons for issues selected (with estimates/risks)
- D2-6 | Issue 6286 - PlantType.create(String) NullPointerException — 3, 0/2

Needs Verification (5)
- + Add a card

Under Verification (3)
- D2-6 | Issue 6326 - Remove misleading @Nullable from IForgeBlock::canCreatureSpawn — 2, 2/2

Done
- D2-1 | Issue Investigation (minimum 5)
- D2-2 | Issue Selection (minimum 2)
- D2-6 | Issue 6325 - IPlantable defau SeaGrass and SeaPickle blocks to Plains plant type — 2/2

8) Issue 6326 is verified and moved to done; issue 6286 is finished and requires verification


MinecraftForge | Les Programmeurs Free | Team Visible | Invite

To Do
- + Add a card

Blocked
- D2-6 | Issue 6040 - Custom banner patterns no longer work, additional patches needed — 1
- D2-6 | Issue 6405 - Registering villa types per biome is impossible — 7, 1

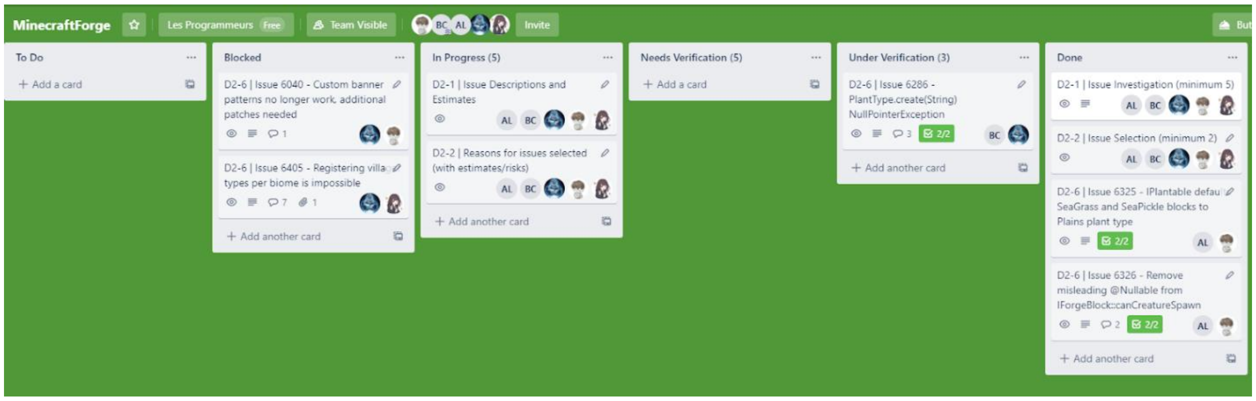
In Progress (5)
- D2-1 | Issue Descriptions and Estimates
- D2-2 | Reasons for issues selected (with estimates/risks)

Needs Verification (5)
- D2-6 | Issue 6286 - PlantType.create(String) NullPointerException — 3, 2/2

Under Verification (3)
- + Add a card

Done
- D2-1 | Issue Investigation (minimum 5)
- D2-2 | Issue Selection (minimum 2)
- D2-6 | Issue 6325 - IPlantable defau SeaGrass and SeaPickle blocks to Plains plant type — 2/2
- D2-6 | Issue 6326 - Remove misleading @Nullable from IForgeBlock::canCreatureSpawn — 2, 2/2

9) Issue 6286 is under verification; write-ups for the issues are still in progress


MinecraftForge | Les Programmeurs Free | Team Visible | BC AL | Invite

To Do
- Add a card

Blocked
- D2-6 | Issue 6040 - Custom banner patterns no longer work, additional patches needed — 1
- D2-6 | Issue 6405 - Registering villa types per biome is impossible — 7 — 1
- Add another card

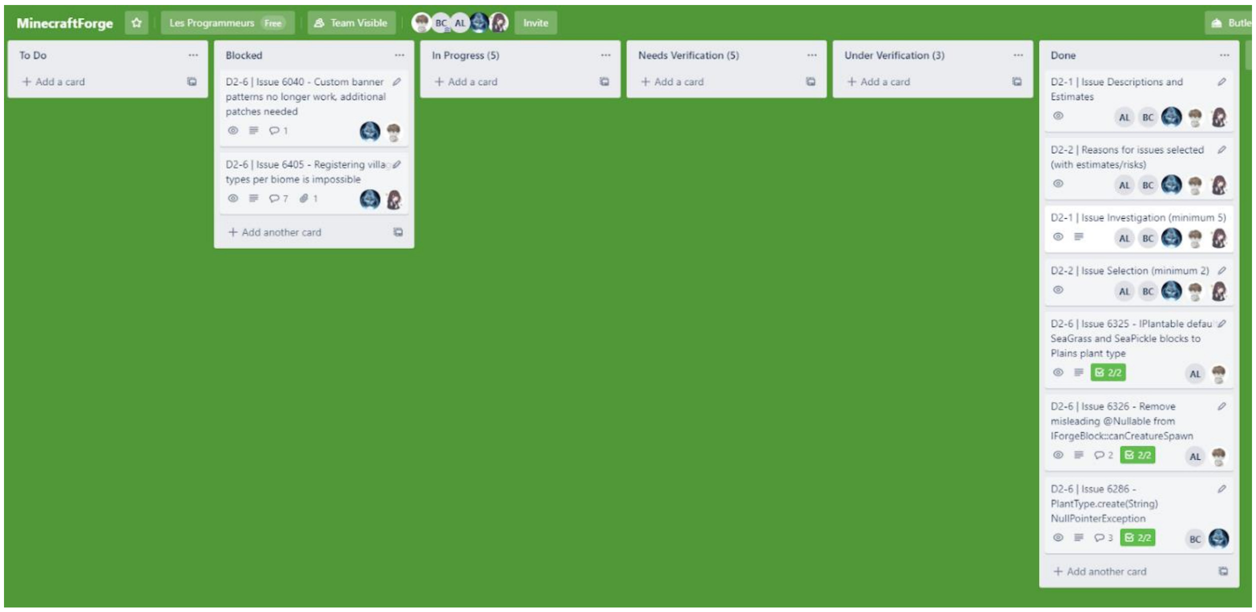
In Progress (5)
- D2-1 | Issue Descriptions and Estimates — AL BC
- D2-2 | Reasons for issues selected (with estimates/risks) — AL BC
- Add another card

Needs Verification (5)
- Add a card

Under Verification (3)
- D2-6 | Issue 6286 - PlantType.create(String) NullPointerException — 3 — 2/2 — BC
- Add another card

Done
- D2-1 | Issue Investigation (minimum 5) — AL BC
- D2-2 | Issue Selection (minimum 2) — AL BC
- D2-6 | Issue 6325 - IPlantable defau SeaGrass and SeaPickle blocks to Plains plant type — 2/2 — AL
- D2-6 | Issue 6326 - Remove misleading @Nullable from IForgeBlock::canCreatureSpawn — 2 — 2/2 — AL
- Add another card

10) Everything is completed.


MinecraftForge | Les Programmeurs Free | Team Visible | BC AL | Invite

To Do
- Add a card

Blocked
- D2-6 | Issue 6040 - Custom banner patterns no longer work, additional patches needed — 1
- D2-6 | Issue 6405 - Registering villa types per biome is impossible — 7 — 1
- Add another card

In Progress (5)
- Add a card

Needs Verification (5)
- Add a card

Under Verification (3)
- Add a card

Done
- D2-1 | Issue Descriptions and Estimates — AL BC
- D2-2 | Reasons for issues selected (with estimates/risks) — AL BC
- D2-1 | Issue Investigation (minimum 5) — AL BC
- D2-2 | Issue Selection (minimum 2) — AL BC
- D2-6 | Issue 6325 - IPlantable defau SeaGrass and SeaPickle blocks to Plains plant type — 2/2 — AL
- D2-6 | Issue 6326 - Remove misleading @Nullable from IForgeBlock::canCreatureSpawn — 2 — 2/2 — AL
- D2-6 | Issue 6286 - PlantType.create(String) NullPointerException — 3 — 2/2 — BC
- Add another card