Project 7 - Final Report
Team Members: Henry Saver and Devin Riess
Project: Terraria Mod - Terraria Expansion of Everything

Deliverables

Github: Link

Note: If you would like to try our code and have access to a PC with Terraria, please reach out! It is very easy to install / run, and we would be more than happy to give a tutorial on this.

Final State of System Statement

Work Done

We implemented most of the features that we laid +out in the beginning of the project.

Devin:

- UI Infrastructure Configuration
- Custom UI Panel C# Object
- Flight Timer C# Object (Observer Pattern)
- Flight Timer UI Addition
- Ammo counter UI element
- Teleporter Item
- Geodude mount item
- Weather changing item
- Commands for the following
  - Finding NPC
  - Load equipment loadout
  - Save equipment loadout
- Pet Stands
  - Dio Item
  - Za World Item

Henry:

- Modded/Custom Items Infrastructure Configuration
- Custom Projectile System (Strategy Pattern w/ Randomization)
- “MegaRock” Custom Projectile Object
- “SuperBoots” Custom Armor Object
- “StaffOfMysteries” Custom Weapon Object

- “MegaRod” Custom Fishing/Tool Object
- Assets for all of the above
- Listener Observer Class for publishing NPC Locational data to the NPCFinder
- Custom Ranged Weapon Class
- Ranged Weapon children of above class (Operator - Sniper, Vandal - Rifle, Spectre - SMG)
- Projectile Decorator Classes for the Custom Ranged Weapons
- AmmoRecipe Factory Class for generating recipes for all our custom items and adding said recipes to the Terraria game (recipes are used to craft our items)

Team:

- Visual Assets + Meshes for the items

Work Not Completed and Changed

NPC Finder

- This was a UI element, we got the common working and left it at that. We didn’t feel the need to add it as UI

Drawing on the Map

- Never got around to implementing this and were less excited about it by the end

Extra equipment/Pet slots

- This was mostly a UI implementation and didn’t show our development in OOP principles

Separating equipment types

- We didn't find this implementation to be necessary for our end goals and it would have wasted time

Loadout saver

- The Loadout saver and loader were changed from a UI element to just a command

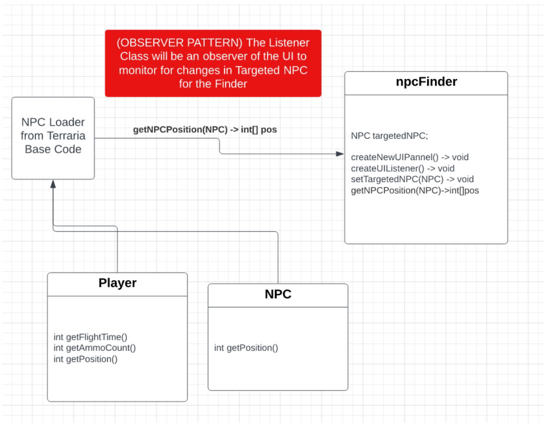
Teleportation Item

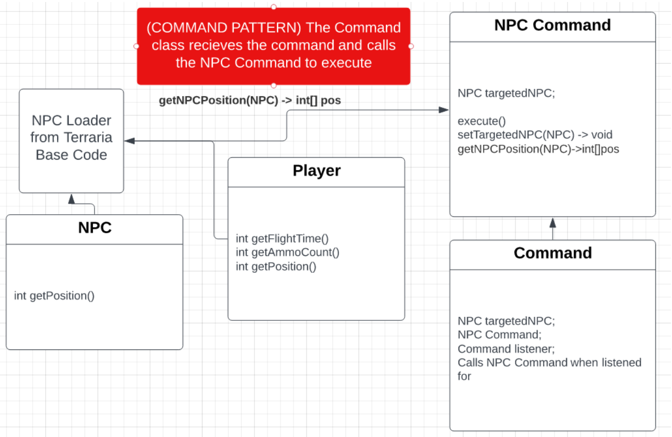
- Changed item to teleport the player to the cursor on the screen, not in the map

Final Class Diagram and Comparison Statement

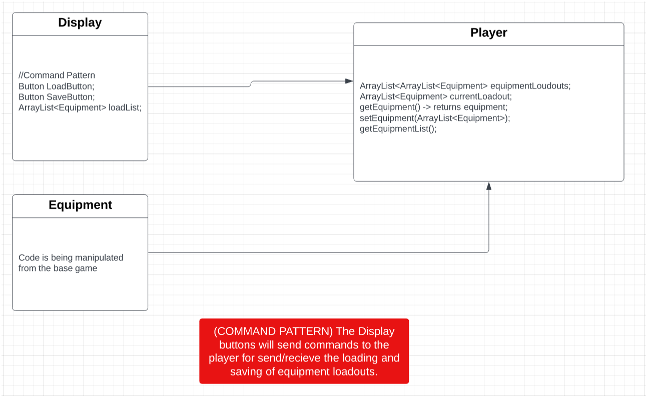
UML Class Diagram & Pattern Use

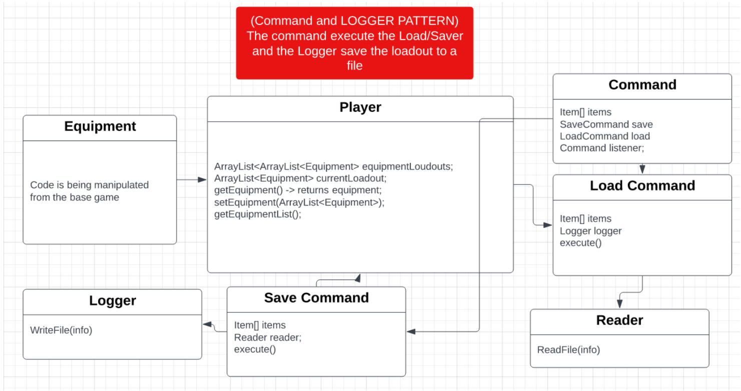
NPC Finder:


(OBSERVER PATTERN) The Listener Class will be an observer of the UI to monitor for changes in Targeted NPC for the Finder
NPC Loader from Terraria Base Code
getNPCPosition(NPC) -> int[] pos
npcFinder
NPC targetedNPC;
createNewUIPannel() -> void
createUIListener() -> void
setTargetedNPC(NPC) -> void
getNPCPosition(NPC)->int[]pos
Player
int getFlightTime()
int getAmmoCount()
int getPosition()
NPC
int getPosition()

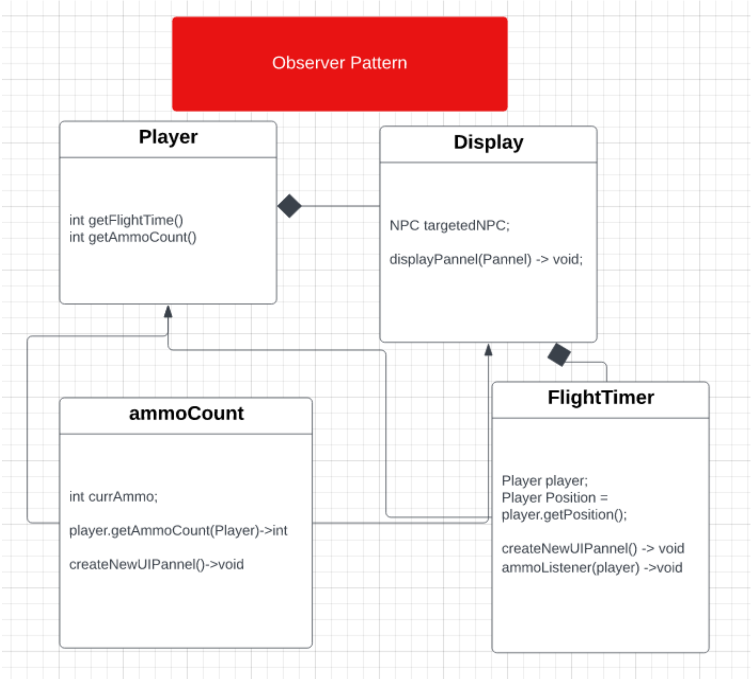

(COMMAND PATTERN) The Command class recieves the command and calls the NPC Command to execute
NPC Command
NPC targetedNPC;
execute()
setTargetedNPC(NPC) -> void
getNPCPosition(NPC)->int[]pos
getNPCPosition(NPC) -> int[] pos
NPC Loader from Terraria Base Code
Player
int getFlightTime()
int getAmmoCount()
int getPosition()
NPC
int getPosition()
Command
NPC targetedNPC;
NPC Command;
Command listener;
Calls NPC Command when listened for

Loadout Saver/Loader:

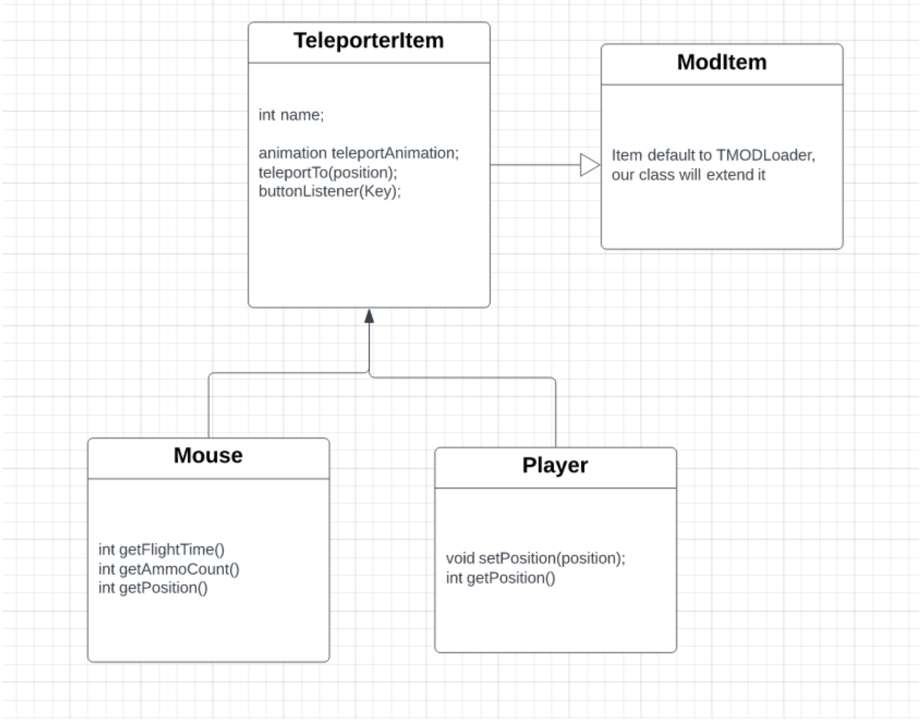

Display
//Command Pattern
Button LoadButton;
Button SaveButton;
ArrayList<Equipment> loadList;
Player
ArrayList<ArrayList<Equipment> equipmentLoudouts;
ArrayList<Equipment> currentLoadout;
getEquipment() -> returns equipment;
setEquipment(ArrayList<Equipment>);
getEquipmentList();
Equipment
Code is being manipulated from the base game
(COMMAND PATTERN) The Display buttons will send commands to the player for send/recieve the loading and saving of equipment loadouts.


(Command and LOGGER PATTERN) The command execute the Load/Saver and the Logger save the loadout to a file
Command
Item[] items
SaveCommand save
LoadCommand load
Command listener;
Player
ArrayList<ArrayList<Equipment> equipmentLoudouts;
ArrayList<Equipment> currentLoadout;
getEquipment() -> returns equipment;
setEquipment(ArrayList<Equipment>);
getEquipmentList();
Equipment
Code is being manipulated from the base game
Load Command
Item[] items
Logger logger
execute()
Logger
WriteFile(info)
Save Command
Item[] items
Reader reader;
execute()
Reader
ReadFile(info)

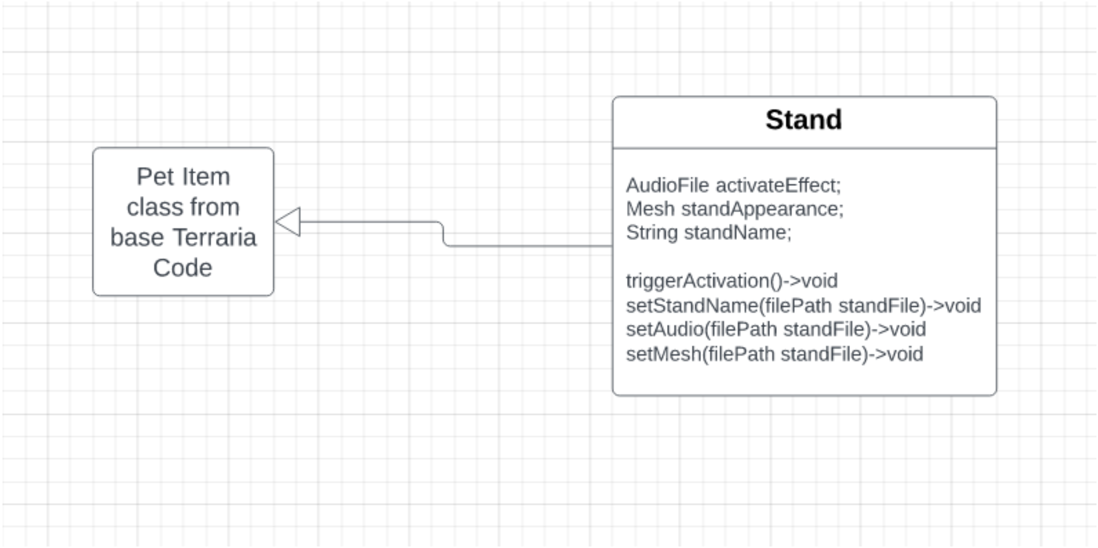
Flight Timer and Ammo Count:



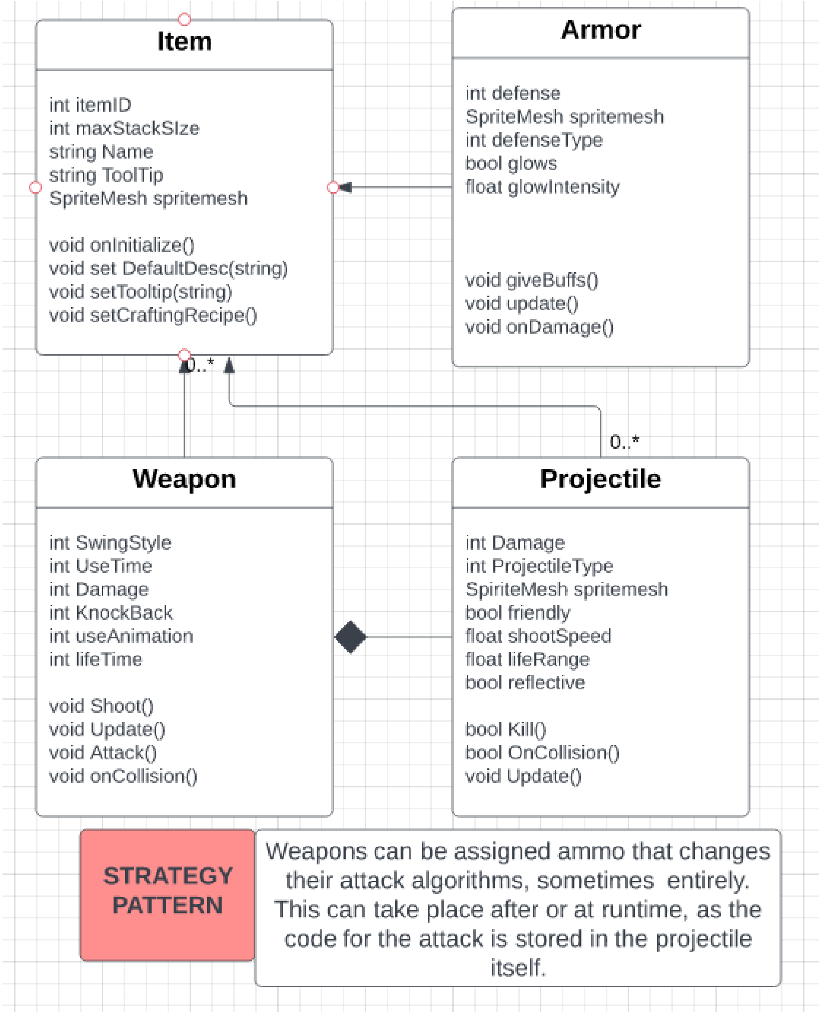
Teleportation Item:



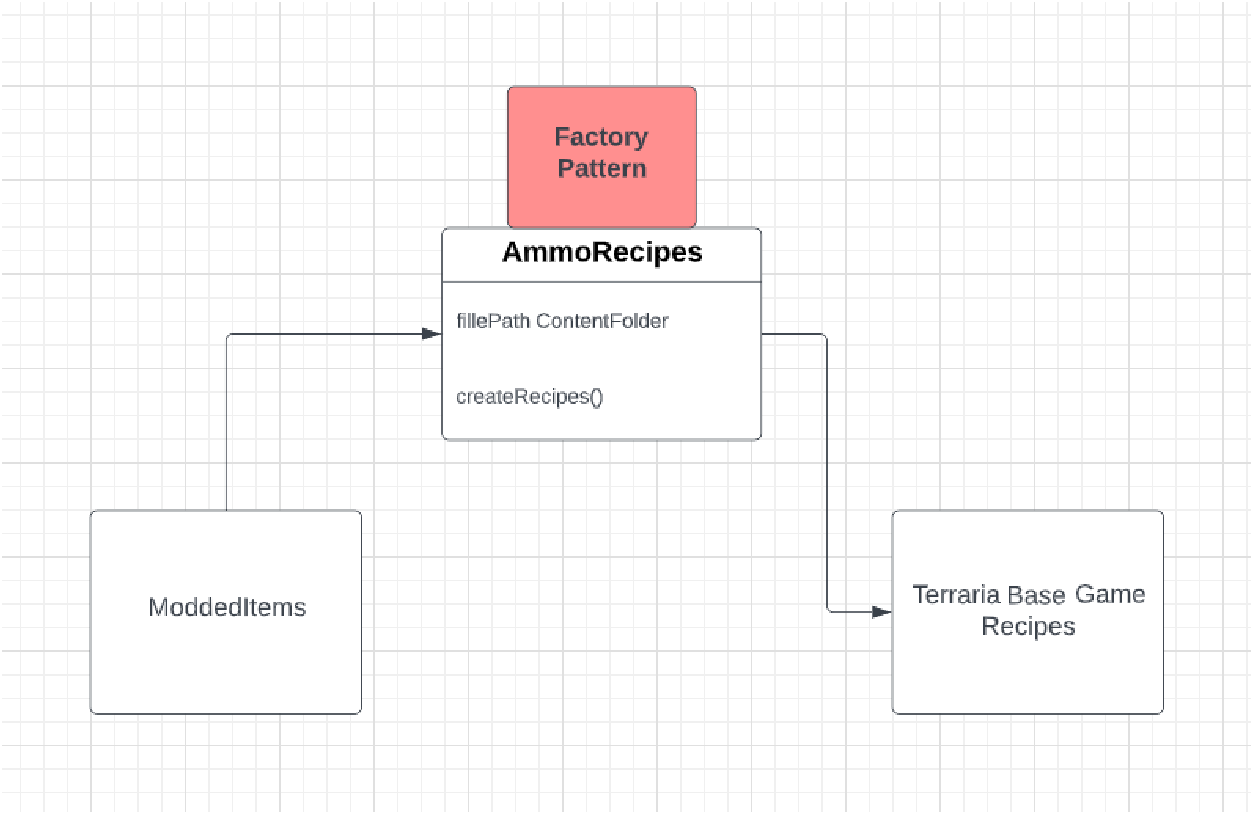
Stand Pet Item:


Pet Item class from base Terraria Code
Stand
AudioFile activateEffect;
Mesh standAppearance;
String standName;
triggerActivation()->void
setStandName(filePath standFile)->void
setAudio(filePath standFile)->void
setMesh(filePath standFile)->void

Magic item and super boots armor Items:



Recipe Factory for generating recipes for our modded items



Changes:

- The npc finder diagram changed a decent amount since we changed the process of the player accessing the NPC finder from a UI element to a command
- The loadout saver/loader pattern was adapted to have a Logger Pattern implemented with it as well as observing the players current equipment loadout
- 2 new items with the magic wand and the super boots were added into the game after project 5 they have not changed since project 6
- Recipe Factory was not initially planned on being added but became a necessity to create once we made important discoveries during development. This was a much more complex process that was relieved by the use of this factory which is why we made the later decision to include it.

Third-Party code vs. Original code Statement

We utilized several pieces of third party software/frameworks to accomplish our goals in this project. They include:

- tModLoader - Mod loading software for injecting our code into Terraria's compilation
- Microsoft XNA Framework - System Calculations and UI compilation (screen size, resolution variables etc)
- tModLoader example Mod - Used as a tutorial for the team to learn Terraria Modding

- Example Resource Bars Mod - Open Source Tutorial for adding custom UI bars
- Valorant Visual Assets - Used as pixelated parody assets for the UI elements for our 3 custom firearm objects
- Terraria Boots Visual Asset - Modified the coloring of the Terraria source shroomite leggings for my creation of the “Supa Boots”

Statement on the OOAD process for your overall Semester Project

Documentation

Throughout the Semester Project, we were dealing with code that has been written and re-written multiple times over the course of Terraria’s lifespan which is over a decade-old game. Due to this as well as it being a game on the smaller side there was not a lot of documentation or example on how one could mod the game. The biggest help throughout this process was TModLoader, the defacto software that lays on top of terraria to allow players to create and load mods into the game. While TModLoader has documentation here, there is no actual description inside any of the classes that define what they do or how they are used. While this slowed down our development process significantly it also gave us a good idea of how vital good documentation and commenting code is to help yourself in the future and other developers who might come along and try to read your code. The example of documentation that help us the most was tmodloader’s examples mod, which granted is a couple of years old but helped us significantly in understanding the basics of how to write a mod (example mod).

Enjoyability

Bouncing back from a negative of the poor documentation the process of creating this mod and seeing code we wrote in one of our favorite games made the whole project worth it. Not only do we enjoy this game thoroughly we added some easter eggs for our friends who also like the game. For example, the Stands, Star Platinum, and Za World from JoJo’s Bizzar adventure made their way into the game because it was a show loved by our friend group. Also one of our friends requested that geodude from pokemon be added so we added him as a rideable mount. Being able to use our creativity was one of the best parts of this project and one of the reasons I appreciate the class allowing us to explore this opportunity.

Software Structure

We got the opportunity to really open the engine room of a AAA title video game that was produced by a massive company, which was a very positive experience for the team. We got to learn about how encapsulation was used in project structuring with different features of the game belonging to different file hierarchies so that developers would know what came from where and what the object was supposed to do. In this sense, Terraria was very well designed because it left room for other features to be added via modding, as their code was very loosely coupled. Most game items and objects could be removed or replaced by modified or custom ones as long as they returned the appropriate signatures. This really made designing a lot of the mod pretty straightforward and allowed us to focus on designing the core components of the mod. This showed us first hand how important this can be for the lifespan of software because Terraria has managed to stay near the top of Steam popularity even being a decade old, by making their code accessible to third party content developers.