Potion Combat Design Doc

This document is intended to outline how combat should work in Cauldron Chronicles. This includes controls, attack types and how the player will interact with enemies and the environment during combat scenarios.

TABLE OF CONTENTS

1 - Combat Overview

1.1 - Intent

The intent of combat in Cauldron Chronicles is to provide players with the ability to deal with occasional hostile creatures in the open world by making use of their crafted potions. This combat system focuses around being more about puzzle solving and using fun interactions to neutralise threats rather than twitch reflexes and quick thinking.

1.2 - Attack Types

To keep combat simple, potions should be equipped to the player. All bottles function the same way and the player can perform two actions while they are equipped; throw or drink.

Throwing a potion will cause the bottle to break on impact and cause the effects of the bottle to splash on enemies in the affected area. Depending on the potion being thrown, this could result in a mist that applies a persistent effect in an area for a few seconds, a splash that applies an effect once in an area or an effect that applies to a single target hit by the potion.

Drinking a potion will grant players the effect of the potion after a short drinking animation. The benefits of drinking a potion over smashing it on the ground and gaining the effect yourself is:

- Drinking is faster than throwing.
- Beneficial effects will only be applied to the player and not nearby enemies.
- Some potions behave differently depending on whether they are consumed or thrown.
- Conservation of potion bottles that can be used later, either as a weak throwable or for crafting.

1.3 - Controls

1.3.1 - Keyboard & Mouse

While the player has a potion equipped, holding left-click will cause the player to wind up a potion throw and throw with the button is released. Using right-click would cause the player to instead drink the potion.

1.3.2 - Controller

2 - Combat Potions

Potions fall into different categories that all fulfill different purposes. In most scenarios, players would typically stick to using a potion in a way that reflects its assigned category, however, we can intentionally design scenarios where players can get creative and break out of these categories or combine effects for interesting results.

2.1 - Empty bottles

Empty bottles can be carried around or are added to the player's inventory when they consume a potion. An empty bottle can be equipped the same way other potions can be and can be thrown at enemies to stun them. This creates an inexpensive way to briefly neutralise an enemy but does not deal damage. Trying to drink from an empty bottle causes the player to do a short animation where they try to drink and then realise it is empty (similar to trying to drink an empty Estus Flask in a Souls game). This animation is short and is there for flavour, rather than existing to punish the player.

2.2 - Offensive Potions

Offensive potions are used to deal direct damage or apply damaging effects to the target. When thrown, it'll typically apply an effect in the area the bottle smashed in. If the player drinks a damaging potion, it'll either apply the effect directly to the player or create a version of the effect concentrated on the player's location. Players would generally throw these potions.

Examples of offensive potions:

- **Explosive potion:**
  - Thrown: Creates a fiery explosion in the area that damages anything in it.
  - Consumed: The player is set on fire, takes damage over time and applies burning to nearby enemies.
- **Poison potion:**
  - Thrown: Creates a poison vapour that poisons enemies that walk through it.
  - Consumed: The player is poisoned.
- **Force potion:**
  - Thrown: Creates a pulse that damages and knocks targets away from the area.
  - Consumed: Damages the player and knocks away nearby targets.

2.3 - Defensive Potions

Defensive potions are used to protect or heal the target. These function the same as offensive potions when consumed or thrown except they usually apply beneficial effects to the target. Players would generally drink these potions.

Examples of defensive potions:

- **Stoneskin Potion:**
  - Thrown: Turns the target's skin to stone, increasing their defense.
  - Consumed: The player's skin turns to stone, increasing their defense.
- **Antidote:**
  - Thrown: Cures poison from targets and eliminates noxious effects in the area.
  - Consumed: The player is cured of poison.
- **Health Potion:**
  - Thrown: Heals any target in the area.
  - Consumed: Heals the player.

2.4 - Utility Potions

Utility potions are more niche in how they function. They can also apply effects by being thrown or drank however, the effects are more situational and are not objectively beneficial or detrimental.

Examples of utility potions:

- **Teleportation Potion:**
  - Thrown: Teleports affected targets towards where they are facing.
  - Consumed: Teleport the player to where they are looking.
- **Bottled Sunlight:**
  - Thrown: Creates a blinding flash that stuns enemies in the area.
  - Consumed: The player's skin emits a glow, lighting up the area.
- **Antigravity Potion:**
  - Thrown: Targets in the area begin to float away for a few seconds.
  - Consumed: The player begins floating. Just be careful you're not too high up when the effect ends!
- **Shrink Potion:**
  - Thrown: Targets in the area are shrunk to a fraction of their size.
  - Consumed: The player is shrunk to a fraction of their size.
- **Slime Vial:**
  - Thrown: A neutral slime spawns at the location that attacks anything nearby.
  - Consumed: The player dies or feints.
- **Bottled Corruption:**

    - Thrown: Corrupts the area, instantly altering flora and fauna in the area into their corrupted version.
    - Consumed: The player is corrupted, taking damage over time until the corruption is cleansed and corrupting the area around them.

2.5 - Depth Through Player Creativity

While potions have an obvious use on the surface, for example, health potions heal when consumed, explosive potions blow up enemies then thrown etc. much of the depth of this system comes into play when using potions in creative ways. A few examples of this could be:

- **Antidotes:**
    - Throwing an Antidote at a creature emitting a poisonous fog, disabling the fog and letting the player safely pass or harvest precious materials.
    - Throwing an Antidote at a sickly deer, causing it to thank you by dropping “Nature’s Bounty” or some other useful material.
- **Stoneskin Potion:**
    - Throwing a Stoneskin potion at an enemy that is impervious to your other potions. Now that it is made of stone, you may have another potion that can deal with stone foes.
    - Giving yourself Stoneskin to traverse a poisonous swamp safely.
- **Force Potion:**
    - In a situation where you are surrounded, drinking a force potion may be the preferred way to use this as it would knock back enemies around you.

3 - Enemies

The key to making this combat system fun is having interesting enemies that react in fun ways to the effects of the potions. The purpose of this design document is not to design specific enemies but will touch on how enemies and bosses would play with it.

3.1 - Enemy Encounters

At its core, Cauldron Chronicles is meant to be a cosy, accessible game without high-stress combat scenarios. As such, most of the enemies present in the open world would be rather straightforward to deal with some types requiring light strategy to defeat. Enemy encounters in the open world are also largely optional and the player can choose to avoid or leave a combat encounter if they are not prepared.

When players reach later biomes or dungeons, we can expect more of them but still should not be making overly difficult combat encounters. Instead, we can reward players who properly prepare for certain enemy types with satisfying encounters and useful

rewards, for example, bringing fire damage potions into an ice biome allows you to easily defeat stronger ice biome enemies for powerful ice-themed materials.

3.2 - Boss encounters

Boss encounters are an opportunity to challenge the player and should be more difficult than the enemies seen in the area. I would like to split boss encounters into two categories:

- Open-World Bosses:
    - These bosses patrol areas of the map and don't have a set arena.
    - They would require some preparation and skill to overcome but function like a stronger version of the other enemies in the area.
    - Their mechanics would be similar to those of enemies in the surrounding area.
        - For example, if players deal with lots of poison in an area, then the boss fight would also include poison and would be defeated in a similar way to enemies in the biome.
    - In some cases, they could be looked at as the biggest challenge a biome has to offer and could be used as a narrative tool to encourage the player to move to a new area. (similar to Valheim)
- Dungeon Bosses:
    - These bosses would be placed in unique arenas that support the gameplay and strategies that players would use against them.
    - They should be viewed more as a hard puzzle with hazards than an Elden Ring boss.
    - They would provide a challenge to players and the rewards should be substantial but they are not mandatory to progress the game.

4 - Environment

4.1 - Environmental Interactions

In the open world, we can allow some objects to interact with the player's potions and create meaningful environmental interaction. For instance, we could have areas with steep cliffs players can knock enemies off or perhaps a Force Potion could knock a boulder or log into enemies.

Dungeons are an opportunity to provide players with more curated and challenging combat scenarios. Dungeon layouts can be designed to accommodate certain strategies, for example, having a dungeon with chasms that can be abused by using Antigravity or Force potions to displace enemies or traps that players can avoid by making use of a Shrink Potion.

4.2 - Corruption

I gave the example of having a Corruption Potion above but wanted to expand on this here. A Corruption (and potentially Anti-Corruption) potion can allow the player to alter a combat scenario by either spreading or dispelling corruption in an area.

4.3 - Biome interactions

Depending on the biome a player is in, certain potions could be more or less potent. For example, in a Volcanic region, enemies may be more resistant to fiery potions unless they have been weakened or changed in some way.

5 - States & Status Effects

Cauldron Chronicles juggling of effects has the potential to be extremely complex and bloated so this system is designed to be scalable while keeping things relatively simple for developers and players to understand.

5.1 - Player & Enemy States

Players and NPCs have a set of states that is checked whenever an effect is applied to them which will then produce an effect depending on the state of the player. An important part of this system involves moving away from tracking health as a numerical value and instead treating it as the following states:

- Healthy
- Injured
- Down

Any effect that is tagged as damage will replace the Healthy state with the Injured state or the Injured state with the Down state. When the player has the Down state, they are returned to the hut and suffer a penalty for dying, be it dropping all or a portion of carried resources (to avoid players using dying as a means of fast travel).

5.2 - Status Effect Interactions

Status effects are tracked in the same way as health states but would also have a duration assigned to them. A few examples of standard status effects would be:

- Stun: The target is briefly stunned.
- Burning: The target is lit on fire and suffers damage over time. Can be cured by making contact with water.

- Poison: The target is poisoned and suffers damage over time. Can be cured by an antidote. The target cannot heal while poisoned.

As a simple example, let's say the player stands in a fire and has the Burning status applied to them. On the character, we react to this status by turning on a VFX. After 4 seconds, the Burning status is removed and the player goes from the Healthy state to Injured.

Now let's say the player walks into water, the character would have a Wet status applied to them that would look for the Burning state and remove that from the player which would in-turn turn off the Burning VFX and stop the player from losing a Health state.

Similarly, if the player has just come out of the water and is still wet when stepping into the fire, we would attempt to apply the Burning status but it would check for any statuses that counter it, find the Wet status and fail to apply.

5.3 - Non-Standard Interactions

On the standard parent character, I would imagine we would include the reaction logic to each state that a target would typically have to avoid having to duplicate code and efforts. On characters that would have a unique interaction with a specific effect, we would then do something more custom.

For example, let's 90% of enemies in the game have a typical reaction to fire while in the Injured state which would be:

1. The Burning status is applied unless a counter status like Wet or Stoneskin is applied.
2. A Burning VFX is switched on.
3. After X seconds, the target becomes Injured.
4. The status is removed and the VFX switches off.

However, let's say we have a Fire Elemental that instead heals from the Burning state. We would instead override the parent character's reaction event and do the following:

1. The Burning status is applied unless a counter status like Wet or Stoneskin is applied.
2. A Burning VFX is switched on.
3. After X seconds, the target becomes Healthy.
4. The status is removed and the VFX switches off.