Table of Contents

Project Folder Structure:

Assets

- Scripts
  - Overworld
  - Combat
  - UI
  - Characters
  - Items
  - Utilities
- Scenes
- Art
  - Sprites
  - Animations
  - UI
- Audio
  - Music
  - SFX
- Prefabs

- Materials

Important Values:

Damage equation:

(attackStat * (randVal(0.8,1.2)) * attackPower * CritMultiplier * BlockMultiplier * DefenseMultiplier * WeaknessMultiplier * Heat Multiplier * IgniteMultiplier)

- attackPower: 0.6 for the action “attack”, unique for spells
- CritMultiplier: 1.25
- BlockMultiplier: 1 - (0.5)
- DefenseMultiplier: 1 - ((defense * percentReduction) - flatReduction)
    - A character is vulnerable while they are overheated
    - Vulnerable:0.5
- WeaknessMultiplier: 1.5
- HeatMultiplier: 1 + selfHeatMultiplier + oppositeHeatMultiplier
    - 0.2 for both
- IgniteMultiplier: 1+fuel*0.4

Effectivestat equation:

((BaseStat * statAdjustment) * (1 + sum of Percentage Additives) + (sum of Flat Bonuses))

- Attack
- Defense
- Health
- Speed
- Mp
- Crit rate
- Block rate

Max Level

50

LevelUp Equation

ExperienceRequiredForNextLevel=CurrentLevel^(2)×10

Freeze or Burn

freeze: lock them in place for 2 more turn
burn: lock 1 turn and decrease defenses

Stat Growth Ideas:

Low: 5-10
Moderate: 11-20
High: 21-30
Very High: 31-40

Special cases:

- Growths
    - quartered for defence, mp, crit rate, block rate
    - 1.5x for hp

MC: Moderate for all

Storm (Destruction, AoE, Red)

HP: Moderate
MP: High (to facilitate frequent AoE attacks)
Attack: High
Defense: Low (Glass cannon)
Speed: Moderate
Crit Rate: High (Explosive and unpredictable)
Block Rate: Low

Lightning (Rage, Single, Purple)

HP: Low
MP: Moderate
Attack: Very High (High single-target burst)
Defense: Moderate
Speed: Very High (Strike swiftly)
Crit Rate: Very High (Focused strikes aiming to devastate single targets)
Block Rate: Low

Sun (Light, Healing, Yellow)

HP: Moderate
MP: Very High (Consistent healing)
Attack: Low
Defense: High (Survivability is key)
Speed: Moderate
Crit Rate: Low (More focused on healing than damaging)
Block Rate: Moderate (Ensures survival to continue healing)

Rain (Purification, Buff, Blue)

HP: Low (weak to hits in battle to provide buffs)

MP: High
Attack: moderate
Defense: Low
Speed: High
Crit Rate: Low (Not focused on damage)
Block Rate: High (Protection while providing buffs)

Snow (Embers,Debuff,white):

HP: Moderate to High (To take hits and continually apply debuffs)
MP: High (Frequent use of cold-themed debuffs)
Attack: Moderate
Defense: High (They're resilient like a snowstorm)
Speed: Low to Moderate (Cold slows things down)
Crit Rate: Moderate (While not their primary function, the cold can be piercing at times)
Block Rate: High (Resilient like the endurance of winter)

Cloud (Spread, rampingDamage, green)

HP: Moderate
MP: High (Their skills often have multiple turns of effects, requiring more mana)
Attack: Moderate (Their attacks aren't instant but spread over time)
Defense: Moderate
Speed: Moderate
Crit Rate: Moderate (Their damage grows over time, rather than delivering critical blows)
Block Rate: Moderate

Sky(only for spells)

BaseStats

Clone Base Stats:

- HP: Moderate (15)
- MP: Moderate (15)
- Attack: Moderate (15)
- Defense: Moderate (15)
- Speed: Moderate (15)
- Crit Rate: Moderate (15%)
- Block Rate: Moderate (15%)

Main Character Base Stats:

- HP: High (25)
- MP: High (25)
- Attack: High (23)
- Defense: High (22)
- Speed: Moderate (18)
- Crit Rate: Moderate (18%)
- Block Rate: Moderate (18%)

MC Adjustments

- **Storm**
    - HP Multiplier: 0.9x (Slight decrease as a trade-off for high AoE power)
    - MP Multiplier: 1.2x (More mana for frequent AoE skills)
    - Attack Multiplier: 1.25x (Boost to the already high base attack)
    - Defense Multiplier: 0.8x (Glass cannon concept)
    - Speed Multiplier: 1.1x
    - Crit Rate Multiplier: 1.2x (Higher chance to crit with AoE attacks)
    - Block Rate Multiplier: 0.9x (Slightly reduced defensive capabilities)
- **Lightning**
    - HP Multiplier: 0.85x (They trade health for high single-target damage)
    - MP Multiplier: 1.1x
    - Attack Multiplier: 1.3x (Significant boost for high single-target damage)
    - Defense Multiplier: 1x
    - Speed Multiplier: 1.4x (Due to their nature of striking swiftly)
    - Crit Rate Multiplier: 1.4x (High burst potential on single targets)
    - Block Rate Multiplier: 0.9x
- **Sun**
    - HP Multiplier: 1.1x (Increased survivability)
    - MP Multiplier: 1.3x (High mana for consistent healing)
    - Attack Multiplier: 0.9x (Lesser focus on attack)
    - Defense Multiplier: 1.2x
    - Speed Multiplier: 1.0x (Standard speed, acts in the middle of the turn order)
    - Crit Rate Multiplier: 1x (Standard crit rate, as primary focus isn't on dealing damage)
    - Block Rate Multiplier: 1.2x (Enhanced defense due to its healing role)
- **Rain**
    - HP Multiplier: 0.9x
    - MP Multiplier: 1.25x (For frequent buff skills)
    - Attack Multiplier: 1x
    - Defense Multiplier: 0.8x
    - Crit Rate Multiplier: 1.1x (Benefit from buffs, which might enhance attacks slightly)
    - Speed Multiplier: 1.2x
    - Block Rate Multiplier: 1.1x
- **Snow**
    - HP Multiplier: 1.15x (To withstand hits)
    - MP Multiplier: 1.2x (For frequent debuff skills)
    - Attack Multiplier: 1x
    - Defense Multiplier: 1.25x
    - Speed Multiplier: 0.9x
    - Crit Rate Multiplier: 0.9x (Focus is on debuffs rather than high damage)
    - Block Rate Multiplier: 1.3x (Strong defensive potential)
- **Cloud**
    - HP Multiplier: 1.1x

  - MP Multiplier: 1.2x (Frequent spread damage skills)
  - Attack Multiplier: 1.1x
  - Defense Multiplier: 1x
  - Speed Multiplier: 1x
  - Crit Rate Multiplier: 1.2x (Frequent spread attacks have a decent chance to crit)
  - Block Rate Multiplier: 1x (Standard block rate)

SpellList

Storm

Whirlwind Strike

- name: "Whirlwind Strike"
- manaCost: 12
- basePower: 0.9
- numHits: 1
- spellType: Storm
- effectType: SINGLE_TARGET_DAMAGE
- Description: Strikes a single enemy. If the enemy is under the "Gusted" state, the damage spreads to adjacent enemies and removes the "Gusted" state from the primary target.
- Animation: basic projectile

Gusting Blow

- name: "Gusting Blow"
- manaCost: 10
- basePower: 0.7
- numHits: 1
- spellType: Storm
- effectType: AOE_DAMAGE
- Description: Sends a gust to all enemies, increasing speed for every “Gusted” enemy hit.
- ANIMATION:whip like slash of wind

Cyclone Pull

- name: "Cyclone Pull"
- manaCost: 15
- basePower: 1.25
- numHits: 1
- spellType: Storm
- effectType: ADJACENT_AOE_DAMAGE
- Description: Creates a cyclone pulling and damaging adjacent enemies. If they are under "Gusted", the damage is increased.
- Animation: ORB with sucking in particle

Tempest Surge

- name: "Tempest Surge"
- manaCost: 8

- basePower: 0
- numHits: 0
- spellType: Storm
- effectType: BOOST
- Description: Boosts the character's attack, for the next turn, restore mana based on damage dealt
- Animation: we'll give buffs the same animation different color

Typhoon's Wrath

- name: "Typhoon's Wrath"
- manaCost: 30
- basePower: 1.75
- numHits: 1
- spellType: Storm
- effectType: AOE_DAMAGE
- Description: Summons a typhoon that damages all enemies. if an enemy that is "Gusted" is killed returns to the top of the turn wheel.
- Animation : Flame tornado

Lightning

Voltage Jab:

- name: Voltage Jab
- manaCost: 6
- basePower: 1
- numHits: 1
- spellType: Lightning
- effectType: SINGLE_TARGET_DAMAGE
- Special: Each non-critical Voltage Jab increases the crit chance by 10%, up to 30% for 5 turns
- Description: A swift electrical punch.
- Animation: basic projectile- maybe make it thin out should definitely be fast

Charged Pyke:

- name: Charged Lunge
- manaCost: 10
- basePower: 1.3
- numHits: 1
- spellType: Lightning
- effectType: SINGLE_TARGET_DAMAGE
- Description:Strikes a sharp blow from below with electrical energy.
- Animation: lighting pillar comes out from under them

Static Buildup:

- name: Static Buildup
- manaCost: 12
- basePower: 0.8

- numHits: 1
- spellType: Lightning
- effectType: SINGLE_TARGET_DAMAGE
- Specia: A surging electric field that when crits makes the enemy "Static Charged", more vulnerable to critical hits.
- Description: An attack that has a chance to make the target more vulnerable to critical hits.
- Animation: orb with electric lines coming off of it

Precision Bolt:

- name: Precision Bolt
- manaCost: 18
- basePower: 1.5
- numHits: 1
- spellType: Lightning
- effectType: SINGLE_TARGET_DAMAGE
- Special: If the target is "Static Charged", this spell will remove the "Static Charged" state and guarantee a critical hit.
- Description: A carefully aimed bolt of electricity that exploits static charges on the enemy.
- Animation: electric spear

Thunderstrike:

- name: Thunderstrike
- manaCost: 40
- basePower: 3
- numHits: 1
- spellType: Lightning
- effectType: SINGLE_TARGET_DAMAGE
- Special: If the target's HP is less than 20%, or if they are "Static Charged", this spell guarantees a critical hit.
- Description: An overwhelming electric assault, especially potent against weakened or static charged foes.
- Animation: swirling fire, above, strike down below and a fire’ie impact

Sun

Glowing Embrace:

- name: "Glowing Embrace"
- manaCost: 10
- basePower: 1.5 (Multiplier for healing amount)
- numHits: 1
- spellType: Sun
- effectType: HEAL
- description: Heals a single ally.
- Animation: heal animation, based on fire, sparkles

Luminous Shield:

- name: "Luminous Shield"

- manaCost: 15
- basePower: 0 (Doesn't do damage or heal directly)
- numHits: 1
- spellType:Sun
- effectType: BUFF
- description: Places a protective shield around an ally, absorbing a fixed amount of attacks. When the shielded ally is attacked, a portion of the damage is reflected back onto the attacker.(**IMPORTANT** Made it just a defense buff since it's easier)
- Animation: shield piece in two connecting

Radiant Pulse:
- name: "Radiant Pulse"
- manaCost: 20
- basePower: 1.2 (Multiplier for healing amount)
- numHits: 1
- spellType:Sun
- effectType: EffectType.AOE_HEAL
- description: Heals all allies.
- Animation: heal animation

Sun Mark:
- name: "Sun Mark"
- manaCost: 25
- basePower: 0 (Doesn't do damage or heal directly)
- numHits: 1
- spellType: Sun
- effectType: DEBUFF
- description: Sun Marks a single enemy, lowering their defenses for a set number of turns.
- Animation: default debuff yellow

Solar Salvation:
- name: "Solar Salvation"
- manaCost: 50
- basePower: 2.5 (Multiplier for healing amount)
- numHits: 1
- spellType: FireType.SUN
- effectType: AOE_HEAL
- Effect: Greatly heals all allies and revives any that are downed. Damages sun marked enemies for 10% max health ignoring damage calc
- Animation:Light ray from above that shines on the battlefield, creating a flash on enemies and heal animation on allies

Rain

Mist of Renewal:
- name: "Mist of Renewal"
- manaCost: 10

- basePower: 1
- numHits: 1
- spellType: FireType.RAIN
- effectType: EffectType.HEAL
- description: Heals a single ally slightly.

Mist Barrage:

- name: "Mist Barrage"
- manaCost: 15
- basePower: 0.2
- numHits: 5
- spellType: FireType.RAIN
- effectType: EffectType.MULTIHIT_TARGET_DAMAGE
- Effect:Summons a series of mist projectiles that strike a single enemy target multiple times. heals a random ally a small amount.

Downpour Dance:

- name: "Downpour Dance"
- manaCost: 25
- basePower: 0
- numHits: All allies
- spellType: FireType.RAIN
- effectType: AOE_BUFF
- description: All allies' spells now also have the Rain fire type for a few turns, making their attacks incorporate Rain attributes and one random attribute that an enemy on the field is weak too.

Cascade Clarity:

- name: "Cascade Clarity"
- manaCost: 20
- basePower: 0
- numHits: 1 (single target)
- spellType: FireType.RAIN
- effectType: EffectType.BUFF
- description: Targets one ally. For the next few turns, this ally's spells have a reduced mana cost (let's say by 20%). Additionally, the ally's speed is increased, ensuring they get more turns to capitalize on the reduced spell costs.

Rainmaker's Fury:

- name: "Rainmaker's Fury"
- manaCost: 40
- basePower: 0
- numHits: All allies
- spellType: FireType.RAIN
- effectType: EffectType.AOE_BUFF
- description: All allies get a significant attack power boost and restores some MP

Snow

Frostbite Fangs:

- name: "Frostbite Fangs"
- manaCost: 12
- basePower: .7
- numHits: 1
- spellType: FireType.SNOW
- effectType: EffectType.SINGLE_TARGET_DAMAGE
- description: Strikes an enemy with ice-coated fangs. Reduces the enemy's speed for their next turn.

Glacial Wall:

- name: "Glacial Wall"
- manaCost: 10
- basePower: N/A
- numHits: N/A
- spellType: FireType.SNOW
- effectType: EffectType.HEAL
- description: Envelops all allies in a protective layer of ice, significantly increasing their defense for a few turns.

Frostlink Chain:

- name: "Frostlink Chain"
- manaCost: 22
- basePower: .8
- numHits: 1
- spellType: FireType.SNOW
- effectType: EffectType.SINGLE_TARGET_DAMAGE
- description: Targets an enemy with a shard of ice. If the target is under any chill, the shard links to another random enemy, copying and refreshing all the debuffs from the primary target to the secondary one.

Frostburn Strike

- Name: Frostburn Strike
- Mana Cost: 20
- Base Power: 0.6
- Number of Hits: 1
- Spell Type: FireType.SNOW
- Effect Type: SINGLE_TARGET_DAMAGE
- Description: Strikes the enemy with a chilling blow that causes lingering frostburn. The frostburn debuff damages the enemy over time, based on the caster's attack stat.

Blizzard Burst:

- name: "Blizzard Burst"
- manaCost: 35
- basePower: 1.4
- numHits: 1
- spellType: FireType.SNOW

- effectType: AOE_DAMAGE
- Effect: Summons a powerful snowstorm on the battlefield, damaging all enemies and significantly reducing their speed for the next turn and applying 3 other random debuffs

Cloud

Breezy Blessing:

- name: "Breezy Blessing"
- manaCost: 10
- basePower: 1.2
- numHits: 1
- spellType: FireType.CLOUD
- effectType: EffectType.SINGLE_TARGET_DAMAGE
  description: Strikes an enemy with a gentle breeze and grants the caster a random buff.

Zephyr Link:

- name: "Zephyr Link"
- manaCost: 15
- basePower: 1
- numHits: 1
- spellType: FireType.CLOUD
- effectType: EffectType.SINGLE_TARGET_DAMAGE
- description: Hits an enemy with wind magic and spreads any active buffs from the caster to allies, refreshing their duration.

Cyclone Surge:

- name: "Cyclone Surge"
- manaCost: 18
- basePower: .8 + (number of buffs on caster * 0.2)
- numHits: 1
- spellType: FireType.CLOUD
- effectType: EffectType.SINGLE_TARGET_DAMAGE
- description: A focused gust that damages an enemy. The more buffs the caster has, the greater the damage.

Mist Mantle:

- name: "Mist Mantle"
- manaCost: 25
- basePower: (number of buffs on caster * .4)
- numHits: 1 per enemy
- spellType: FireType.CLOUD
- effectType: EffectType.AOE_DAMAGE
- Effect: A swirling mist surrounds all enemies, dealing damage relative to the amount of buffs you have(0 buff 0 damage)
-

Tempest's Gift:

- name: "Tempest's Gift"
- manaCost: 35

- basePower: N/A (Utility)
- numHits: N/A
- spellType: FireType.CLOUD
- effectType: EffectType.AOE_BUFF
- description: The entire party is surrounded by a protective tempest, granting each member multiple random buffs. And applies the buff power charge, independent 2x damage

Sky

Horizon's Calm

- Name: Horizon's Calm
- ManaCost: 40
- BasePower: 1.2
- NumHits: 1
- SpellType: Sky
- EffectType: HeatAdjust + DefenseBoost
- Description: Deals damage and adjusts the heat bar to the bottom of the green section. Also, provides a temporary defense boost to the main character.

Celestial Equinox

- Name: Celestial Equinox
- ManaCost: 45
- BasePower: 1.5
- NumHits: 1
- SpellType: Sky
- EffectType: TurnStateChange + AttackBoost
- Description: Deals damage and changes the turn state. Also, provides a temporary attack boost to the main character.

Ascending Dawn

- Name: Ascending Dawn
- ManaCost: 50
- BasePower: 1.3
- NumHits: 1
- SpellType: Sky
- EffectType: TurnOrderManipulation + SpeedBoost
- Description: Deals damage and pushes the entire party to the front of the turn order. Also, provides a temporary speed boost to the main character.

Statuses

buffs

1. **Attack Boost:** Increases the damage dealt by a character. (generic, including Celestial Equinox(Sky),tempest Surge(storm),CascadeClarity(Rain),Rainmakers Fury(Rain))
2. **Speed Boost:** Increases the speed, potentially affecting turn order. (generic,Gusting Speed(Storm),From Ascending Dawn of Sky spells)
3. **Defense Boost:** Reduces damage taken by the character. (generic,From Horizon's Calm of Sky spells,glacial wall(snow))
4. **Mana Buff:** Reduces the mana cost of spells. (Cascade Clarity(Rain))
5. **Luminous Shield:**blocks one attack reflects damage
6. **Downpour Dance:** all attacks now also hit for the rain weakness
7. **Crit rate Boost:** This buff increases the critical hit rate.(Generic,From Voltage Jab)
8. **Power Charge:** damage is 2x
9. **Stratospheric Insight:** Increases the MP regeneration rate.(pulled from clouds random)

10. **Skyward Surge:** A super buff that boosts all stats

Debuffs:
Generics

11. **Attack Down:(**generic)
12. **Speed Down:** (generic,frostbitefang(Snow)blizzard burst(Snow),
13. **Defense Down:**(generic,sunmark(Sun)

Gusted: Amplifies the next storm spell damage and spreads to adjacent enemies.(from storm spells)
Frostburn:damage over time
Static Charged: Increases susceptibility to critical hits. (From Lightning spells)
Sun Marked: Decreases defenses for a duration. (From Sun spells)
Chilled: Reduces the speed of the affected character. (From Frostbite Fangs of Snow spells)
Vulnerable: 50% defense decrease
Freeze: Lock out for 2 turns, vulnerable
Burn: Lock out for 1 turn, vulnerable

Gear:

Any stat flat or percentage increase, 3 tiers

Heat Generation

heatGauge: 0-100
Yellow:40, Green:40, Red:20
Generated heat when using spells: manaCost/5
Generate heat when weakness is hit: 15

Class Structure:

Overworld

Class:CameraController

Description: Handles the camera movement, positioning, and effects for both battles and traversal.

Attributes:

currentMode: CameraMode
    Defines the current mode of the camera: Battle or Traversal.
battlePosition: Vector3
    The position the camera moves to during battles.
traversalPosition: Vector3
    The default position for the camera during traversal/exploration.
followTarget: Transform
    The target (usually the player) that the camera should follow during traversal.
followSpeed: float
    The speed at which the camera follows the target during traversal.
zoomLevel: float
    Current zoom level of the camera.
rotationAngle: float
    Current rotation angle of the camera. Useful for tilting the camera during specific locations or maybe during battle

Methods:

SwitchToBattleMode():
    Transitions the camera to the battle position and adjusts other settings necessary for battle mode.
SwitchToTraversalMode():
    Restores the camera to its default traversal/exploration settings.
Note: don’t implement the below functions for the mvp
FollowTarget():

Moves the camera to follow the specified target, often the player character, during traversal.

Zoom(float amount):

Adjusts the camera's zoom based on the provided amount.

Enums:

CameraMode:
Values: BATTLE, TRAVERSAL

Class: PlayerController

Description: Handles player movement, interactions in the overworld. This controller should interact with the Unity Character Controller for smooth movement.

Attributes:

movementSpeed: float

The speed at which the player moves.

characterController: CharacterController

Unity's built-in character controller component for handling movement.

isInteracting: bool

A flag to determine if the player is currently interacting with something.

currentInteractable: Interactable

Reference to the interactable object the player is currently in contact with (using hitboxes or colliders).

Methods:

Move()

Handles player movement using input**(WASD)** from the player and moves the character using the character controller.

Interact()

If the player is in contact with an interactable object's hitbox or collider, and the interaction button**(Spacebar/Enter)** is pressed, this will trigger the interaction.

Note:attack() is not needed right now

Attack()

Initiates an attack using the player's animation and hitbox to determine hit targets.

HandleCollisionWithInteractable(Interactable interactable)

When the player's collider or hitbox makes contact with an interactable object, this method will set the currentInteractable and potentially display an interaction prompt.

ClearCurrentInteractable()

Clears the currentInteractable reference and hides any interaction prompts.

Class: Interactable

Description:A base class representing objects in the world that the player can interact with, such as NPCs, chests, doors, etc. Interaction is detected based on the player's proximity and direct contact with the interactable's hitbox or collider.

Attributes:

isInteractable: bool
  Specifies if the object is currently interactable.
interactionPrompt: string
  A message or prompt that will be displayed to the player when they approach the object (e.g., "Press E to interact").
interactionIcon: Sprite
  An icon or graphic that can be displayed alongside or instead of the interaction prompt.
interactionCollider: Collider

  The collider (or hitbox) that determines when the player is close enough to interact with the object.

Methods:

OnPlayerEnterInteractionZone()
  Triggered when the player's collider or hitbox enters the interaction zone (the interactionCollider). This can display the interaction prompt and/or icon to the player.
OnPlayerExitInteractionZone()
  Triggered when the player's collider or hitbox exits the interaction zone. This would hide the interaction prompt and icon.
Interact(PlayerController player)
  Defines the action that will take place when the player interacts with the object. This would typically be overridden in derived classes for specific behaviors.
DisplayInteractionPrompt()
  Displays the interaction prompt and/or icon on the UI.
HideInteractionPrompt()
  Hides the interaction prompt and icon from the UI.

Class: Campfire

Extends Interactable
Description:A unique interactable that not only allows players to recover health and replenish healing items but may also serve as a narrative point or checkpoint in the game.

Attributes:

isActive: bool

Represents whether the campfire is currently lit or not. An active campfire can offer its services.

refillAmount: int

The amount by which healing items are replenished when a player interacts with the campfire.

Methods:

Interact(PlayerCharacter player):

Overrides the base Interact method. When a player interacts, it replenishes the player's health and refreshes the healing items.

RefillHealingItems(PlayerCharacter player):

Refills the player's HealingItem count to its maxCount.

LightCampfire():

If the campfire is not active, this will light it, making it usable.

ScriptableObject: Dialogue

Description: The Dialogue class provides an interactable means for players to engage in narrative sequences or receive information. When the player approaches and interacts, a series of text-based messages or exchanges will appear.

Attributes:

dialogueLines: List<string>

Contains the individual lines of dialogue to be displayed in sequence.

currentLineIndex: int

Keeps track of the current line of dialogue being displayed.

isDialogueFinished: bool

Indicates whether the dialogue sequence has reached its conclusion.

npcName: string

The name of the NPC or entity delivering the dialogue. This can be used for UI displays.

portrait: Sprite

An optional portrait or image of the NPC/entity that's speaking. This can add personality and context to dialogues.

Methods:

Interact(PlayerCharacter player):

Overrides the base Interact method. Initiates or progresses the dialogue sequence when the player interacts.

DisplayNextLine():

Shows the next line in the dialogueLines sequence, updating currentLineIndex.

EndDialogue():

Concludes the dialogue sequence, setting isDialogueFinished to true and potentially triggering other game events.

ResetDialogue():

Resets the dialogue's currentLineIndex and isDialogueFinished so it can be replayed, if desired.

Class: NPCController

Extends: Interactable
Description: Handles the specific logic for NPCs, such as initiating dialogues, triggering quests, or other unique NPC interactions.

Attributes:

npcDialogue: Dialogue
Data structure or reference to the dialogue this NPC provides. Could be a series of strings, voice lines, or a more complex dialogue tree.

Methods:

Interact(PlayerController player) Override
Overridden method from the Interactable class. When the player interacts with this NPC, it could initiate a dialogue, offer a quest, etc.
InitiateDialogue()
Starts the dialogue process with the player, displaying text, playing voice lines, etc.

Class: ChestController

Extends: Interactable
Description: Handles the specific logic for chests, such as opening animations, and retrieving items.

Attributes:

isOpen: bool
Indicates whether the chest has already been opened.
containedItems: List<GearItem>
A list of items that are contained within the chest.
chestAnimation: Animator
Reference to the animator component, if the chest has an opening animation.

Methods:

Interact(PlayerController player) Override
Overridden method from the Interactable class. When the player interacts with this chest, it will open the chest and retrieve items.
OpenChest()
Triggers the chest opening animation (if any) and marks the chest as opened.
RetrieveItems()

Gives the contained items to the player, typically adding them to the player's inventory, and then clears the items from the chest.

Combat (Davin)

Class:BattleManager

Description: Responsible for overseeing and managing the turn-based combat logic. It handles the state of the battle, turn orders, win/lose conditions, and combat transitions.

Attributes:

currentBattleState: BattleState
Enum denoting the current state of the battle (e.g., Start, PlayerTurn, EnemyTurn, Won, Lost).

NOTE: player and enemies order in the list needs to align with battle field position, AKA if they are adjacent in the battle they should be adjacent in the list

playerCharacters: List<PlayerCharacter>
List of player characters participating in the battle.

enemyCharacters: List<EnemyCharacter>
List of enemy characters in the battle.

turnQueue: Queue<CharacterBase>
Queue determining the turn order based on character speed or other criteria.

activeCharacter: CharacterBase
The character whose turn is currently active.

lastHitCharacter: CharacterBase
The character who was last attacked

weaknessMultiplier: float
damage multiplier for weakness

playerHeatSystem: HeatSystem
Represents the collective heat gauge for the player's party. Monitors and manages the heat levels based on actions performed by player characters during battle.

enemyHeatSystem: HeatSystem
Represents the collective heat gauge for the enemy party. Monitors and manages the heat levels based on actions performed by enemy characters during battle.

turnState: TurnState
Tracks the current turn state which determines if the secondary effect applied is Freeze or Burn.

fuel: float
Represents the current amount of fuel available. Built up by using the "attack" action and consumed to boost damage.

maxFuel: float
The maximum amount of fuel that can be accumulated.

usedFuel: float
Amount of fuel used for given action

charactersInBattle: List<CharacterBase>

List of all characters (both player and enemy) participating in the battle.

speedThresholdForExtraTurn: float

Speed value above which characters get an extra turn.

Methods:

StartBattle()

Initializes the battle, setting up the initial turn order, and transitioning the game into combat mode.

EndBattle()

Handles the logic once a battle concludes, such as distributing experience, handling narrative events, and transitioning back to the overworld.

NextAction()

Advances the turnWheel, dequeuing the next character in the turnQueue and setting them as the activeCharacter.

ExecuteAction(CharacterBase character, CharacterBase target):

Asks the character for action, and the target then executes it.

CalculateDamage(CharacterBase attacker, CharacterBase target):

Given an attacker and a target, this method will calculate the actual damage based on the provided formula. It will account for crits, blocks and usedfuel (if the attacker is a PlayerCharacter), and any other multipliers in the formula. Will also apply heat based on characters weakness

InitializeTurnOrder()

Sets the initial turn order. Characters' effective speeds are calculated with a base speed and a random modifier. Those with speeds surpassing a threshold might get an extra turn.

AdvanceTurn(): CharacterBase

Dequeues the next character in line for their turn and returns this character to the BattleManager to handle their specific actions.

UpdateTurnThumbnails()

Updates the visual representation of the turn order to give players an idea of upcoming turns.

ReevaluateTurnOrder()

In case of events like a character using a "haste" spell or other speed-altering effects, this reevaluates and adjusts the turn order accordingly.

RemoveCharacterFromTurnOrder(CharacterBase character)

Removes a character from the turn order, such as when they're defeated.

OverHeat():

For the side that just overheated, take the active character(or last hit character if no active character on that side) and set them to overHeated and depending on the turn Freeze or Burn

toggleTurnState():

Switches the turnState between Freeze and Burn. This can be called at the end of each round or turn to alternate the state.

ResolvePreTurnEffects(CharacterBase character):

Called at the beginning of each character's turn to adjust stats or see if the character can act.

ResolvePostActionEffects(CharacterBase character, Action actionTaken):

After a character takes an action, check if any new states need to be applied or if old states expire.

CheckBattleOutcome()

Checks to see if the battle conditions have been met for a win or loss.

ConsumeFuel(float amount):

Decreases the fuel by the specified amount.

AccumulateFuel(float amount):

Increases the fuel by the specified amount, ensuring it doesn't exceed maxFuel.

Nested Enums/Classes:

BattleState

Enum values might include: Start, PlayerTurn, EnemyTurn, Won, Lost, etc.
Action

TurnState

Enum values might include: FREEZE, BURN

ScriptableObject: StatusEffect

Description: Represents a generic status effect that can be applied to a character. Serves as a base class for all specific status effects (both buffs and debuffs).

Attributes:

name: string

The name of the status effect.

duration: int

Number of turns the effect lasts.

source: CharacterBase

The character that applied the status effect.

target: CharacterBase

The character that is affected by the status effect.

Methods:

ApplyEffect():

Abstract method that applies the effect to the target.

RemoveEffect():

Abstract method that removes the effect from the target.

DecreaseDuration():

Decreases the effect's duration each turn.

Nested Classes:

Class: Buff (Inherits from StatusEffect)
Description: Represents a positive effect that is applied to a character.
Attributes:
boostAmount: float
    The amount by which a character's attribute (e.g., attack, defense, speed) is boosted.
Methods:
ApplyEffect(): Overrides the base method to apply the buff to the target.
RemoveEffect(): Overrides the base method to remove the buff from the target.

Class: Debuff (Inherits from StatusEffect)
Description: Represents a negative effect that is applied to a character.
Attributes:
reductionAmount: float - The amount by which a character's attribute is reduced.
Methods:
ApplyEffect(): Overrides the base method to apply the debuff to the target.
RemoveEffect(): Overrides the base method to remove the debuff from the target.

Class:CharacterBase:

Description:A foundational class representing any character in the game. This includes shared attributes and behaviors that both player characters and enemies possess.The CharacterBase class serves as a template for all characters in the game. Specific types of characters, like player characters or different enemy types, would derive from this class and extend it with additional attributes or behaviors unique to that character type.

Attributes:

characterName: string
    The name of the character.
isAlive: bool
    Indicates if the character is currently alive. If currentHealth drops to 0 or below, this becomes false.
characterSprite: Sprite
    Visual representation of the character.
characterStats: CharacterStats
    This will store the base stats for any character, whether it's a player or an enemy
List<StatusEffect> activeStatusEffects:
    A list of currently active status effects on the character.

Methods:

TakeDamage(int damage)

Reduces the character's currentHealth by the given damage value. Checks if the character is still alive afterward.

Heal(int amount)

Increases the character's currentHealth by the given amount, without exceeding maxHealth.

Attack(CharacterBase target):

Directly causes damage to the target based on the character's adjusted attack value.

CastSpell(Spell spell, CharacterBase target):

Deducts the mana cost of the spell from the character's current mana and applies the spell's effect to the target. This method will have to ensure that the character has enough mana to cast the spell.

Act(): Action

Virtual function that determines the action a character will take. Returns an Action.

Die()

Handles the character's death, such as playing a death animation, removing them from the turn order, or triggering specific game events.

ApplyStatusEffect(StatusEffect effect):

Adds the effect to the activeStatusEffects list

RemoveStatusEffect(StatusEffect effect):

removes the effect from from the list.

dNested Enums/Class:

ActionType

Enum values include: ATTACK, IGNITE, USE_ITEM, CAST_SPELL

CharacterAction

Description: A class or struct that wraps up all information related to an action. This includes the type of action (from the ActionType enum), the target (if applicable), and any additional data (like which item or spell).

Attributes:

- **type:** ActionType
- **target:** CharacterBase
- **additionalData:** object (Spell, or other relevant data based on action type)
  - A class or struct that wraps up all information related to an action. This includes the type of action (from the ActionType enum), the target (if applicable), and any additional data (like which spell).

Class: PlayerCharacter:

Extends: CharacterBase

Description: Represents a playable character. This class encompasses attributes and behaviors specific to characters controlled by the player.

Attributes:

playerStats: PlayerStats
  This will store the specific stats for a player character.

Methods:

Attack(CharacterBase target):Override
  Directly causes damage to the target based on the character's adjusted attack value.
Recovers some mp
UseMana(float cost):
  Deducts the cost from currentMana. If currentMana goes below 0, it remains at 0.
RegenerateMana(float amount):
  Adds the amount to currentMana.
UseItem(Item item)
  Use one of the set amount of healing items from the inventory.
Ignite(float amount):
  Uses a specified amount of fuel for the current action. Communicates with the BattleManager to deduct the used fuel.
Act() Override: Action
  This can capture player input through a UI prompt and returns the chosen action.
ChooseTarget(): CharacterBase
  Shows a UI prompt for the player to select a target among available enemies. Returns the chosen target.

Class: EnemyCharacter:

Extends: CharacterBase
Description: Represents a non-playable enemy character. This class encompasses attributes and behaviors specific to adversaries the player will encounter and combat.

Attributes:

aiType: AIType
  Defines the behavior type or strategy the enemy uses in combat. For instance, some enemies might be aggressive, some defensive, others might prioritize using status effects, etc.
enemyStats: EnemyStats
  This will store the specific stats for an enemy character.

Methods:

Act() Override: Action
  Based on the AI behavior type, it determines the best action for the enemy to take. For example, an aggressive AI might prioritize high-damage attacks, while a defensive one might prioritize guarding or healing. Returns an action
ChooseTarget(): CharacterBase

Uses AI logic to select a target from the available player characters. Returns the chosen target.

Nested Enums:

AIType

Enum values might include: AGGRESSIVE, DEFENSIVE, SUPPORT, STATUS_FOCUS, RANDOM, etc.

Class:HeatSystem:

Description: Manages the collective heat gauge for a side (either the player's party or the enemy party) during battle.

Attributes:

currentHeat: float
Current collective heat value for the side.
maxHeat: float
Maximum heat the side can accumulate before one of its characters overheats.
heatLevels: Dictionary<float, HeatThreshold>
Defines various heat levels and their associated thresholds.

Methods:

AddHeat(float amount):
Increases the currentHeat by the specified amount. Checks if any heat level thresholds are crossed and triggers relevant effects.
UpdateHeatMultiplier(): float
Takes your heat and the opposites heat then calculates and returns the heatMultiplier
ReduceHeat(float amount):
Decreases the currentHeat by the specified amount. Checks if any heat level thresholds are crossed and triggers relevant effects.

Characters

SciptableObject: CharacterStats

Description: Base class that encapsulates the fundamental statistical attributes of a character.

Attributes:

baseHealth: float
Base health points without modifiers.
currentHealth: float
Current health points without modifiers.

baseAttack: float
Base attack power without modifiers.
baseDefense: float
Base defensive capability without modifiers.
baseSpeed: float
Base speed without modifiers.
weakness: FireType
The elemental weakness

Methods:

GetStat(StatType type):
Returns the stat after applying all modifiers (to be overridden in subclasses if needed).

Enum StatType:

contains: ATTACK, DEFENSE, HEALTH, CURRENT_HEALTH, SPEED, CRIT_RATE, BLOCK_RATE

ScriptableObject: PlayerStats

Extends CharacterStats

Attributes:

exp: float
Current experience points.
expToNextLevel: float
Experience needed to reach the next level.
level: int
Current level.
baseMana: float
Base mana or magic points without modifiers.
currentMana: float
Represents the character's current mana or magic points.
critRate: float
Base chance of landing a critical hit without modifiers.
blockRate: float
Base chance of blocking an incoming attack without modifiers.
equipmentModifiers: List<StatModifier>
Modifiers applied due to equipped Gear.
elementalType: FireType
The elemental type of the character. Determines the base stats and growths and spells.
currentSpellList: List<Spell>
The list of spells currently available to the character based on their elemental type and level.

equippedIGear: List<Gear>
Gear currently equipped by the player character, organized by equipment slots (e.g., Head, Chest, Weapon).
isClone: bool
True if is a clone, used for for picking stat growths
healthGrowth: float
The amount that health increases upon leveling up.
manaGrowth: float
The amount that mana increases upon leveling up.
attackGrowth: float
The amount that attack increases upon leveling up.
defenseGrowth: float
The amount that defense increases upon leveling up.
speedGrowth: float
The amount that speed increases upon leveling up.
critGrowth: float
The amount that crit increases upon leveling up.
blockGrowth: float
The amount that block increases upon leveling up.
modifiers: StatAdjustment
Instance of the StatAdjusments class which will hold the correction multipliers.

Methods:

GainExp(float amount):
increases the experience by the given amount and checks for leveling up.
LevelUp():
Increases the level and adjusts base stats accordingly based on growths.
CalculateExpToNextLevel():
Calculates the experience needed for the next level.
ApplyEquipmentModifier(StatModifier modifier):
Applies stat changes due to equipment.
RemoveEquipmentModifier(StatModifier modifier):
Removes stat changes from equipment.
ChangeElementalType(FireType type):
Changes the player's elemental type, adjusts the stats, and updates the spell list accordingly.
UpdateSpellList():
Updates the list of available spells based on the character's elemental type and level.
GetEffectiveStat(StatType type):
Returns the effective stat after applying all modifiers (to be overridden in subclasses if needed).

ScriptableObject: StatAdjustment

Description: This class will store multiplier values for the stat adjustments for the main character when they change their current fireType

Attributes:

healthMultiplier: float
  Multiplier for health based on elemental type.
manaMultiplier: float
  Multiplier for mana based on elemental type.
attackMultiplier: float
  Multiplier for attack power based on elemental type.
defenseMultiplier: float
  Multiplier for defense based on elemental type.
speedMultiplier: float
  Multiplier for speed based on elemental type.
critMultiplier: float
  Multiplier for crit rate based on elemental type.
blockMultiplier: float

ScriptableObject: EnemyStats

Extends CharacterStats

Attributes:

expGiven: float
  Experience points given to the player when this enemy is defeated.

Methods:

GetExpGiven():
  Returns the experience points the enemy provides upon defeat.

Class:BattleUI:

Description: Oversees the management and display of all battle-specific UI elements, including health bars, action menus, turn indicators, and more. This class is responsible for updating the visual representation of the battle state, based on information it receives about the current state of the battle.

Attributes:

partyConditionUI: GameObject

UI element(s) showcasing party conditions including health, MP, and any other relevant info.

enemyConditionUI: GameObject

UI element(s) showcasing enemy condition, potentially including health and special status effects.

actionMenu: GameObject

The interactive menu where players choose their actions during their turn.

turnWheel: GameObject

UI representation of the turn order, often showcasing thumbnails or icons of characters and their upcoming turns.

turnStateIndicator: GameObject

Visual indicator that displays the current turn state (Freeze or Burn).

combatReactionsUI: GameObject

UI elements that pop up to show reactions in combat, like damage numbers or special conditions.

Methods:

UpdatePlayerCondition(PlayerCharacter player):

Refreshes the player's UI elements, like health bars or MP, based on the current status of the player character.

UpdateEnemyCondition(EnemyCharacter enemy):

Refreshes the enemy's UI elements based on the current status of the enemy character.

UpdateAllPlayerConditions(List<PlayerCharacter> players):

Iterates through the list of player characters and updates the UI for each one, refreshing elements like health bars, MP, and other relevant information.

UpdateAllEnemyConditions(List<EnemyCharacter> enemies):

Iterates through the list of enemy characters and updates the UI for each one, showcasing their health and any other pertinent status.

ShowActionMenu():

Displays the action menu to the player during their turn.

HideActionMenu():

Hides the action menu.

UpdateTurnWheel(List<CharacterBase> turnOrder):

Updates the turn wheel UI based on the provided turn order.

SetTurnStateIndicator(TurnState turnState):

Adjusts the turn state indicator based on the current turn state (Freeze or Burn).

ShowCombatReaction(string reactionText, Vector2 position):

Displays a combat reaction (like "Critical Hit!" or a damage number) at a specified position on the screen.

Class:OverworldUI:

Description: Manages and updates the UI elements when the player is navigating the overworld. This includes elements like the player's condition, interaction prompts, mini-maps, and other non-battle related UI components.

Attributes:

playerConditionPanel: GameObject
UI elements displaying the player's condition in the overworld, including health, MP, and other resources.
interactionPrompt: GameObject
UI prompt that appears when the player is near an interactable object, indicating how to interact (e.g., "Press E to interact").
questLogIndicator: GameObject
An indicator or list that provides quick info on the player's current story Objective.
isAltHeld: bool
Is true when tab is held, otherwise false

Methods:

Note: after some thought player condition is a bit redundant with the rpgMenu still useful but not necessary make it a low priority
TogglePlayerConditionDisplay(bool isAltHeld):
If isTabHeld is true, it displays the player's condition UI in the overworld. If false, hides it.
ToggleQuestLogDisplay(bool isTabHeld):
If isTabHeld is true, it displays the quest log indicator. If false, hides it.
UpdatePlayerCondition(PlayerCharacter player):
Refreshes the overworld UI representation of the player's condition, like health bars or MP.
ShowInteractionPrompt(Interactable interactable):
Displays the interaction prompt based on the properties of the interactable object the player is near.
HideInteractionPrompt():
Hides the interaction prompt.

Class:RPGMenu:

Description: Manages the entire RPG menu system, ensuring fluid navigation between its various sub-menus and the correct display of player data such as stats, gear, spells, and other game-relevant information.

Attributes:

currentState: UIState
Represents the current panel being displayed.

partyPanel: PartyPanel

Reference to the PartyPanel class which manages the party viewing functionalities.

gearPanel: GearPanel

Reference to the GearPanel class which manages the gear viewing and equipping functionalities.

spellPanel: SpellPanel

Reference to the SpellPanel class which manages the spell viewing functionalities.

itemsPanel: ItemsPanel

Reference to the ItemsPanel class which manages the item viewing functionalities.

optionsPanel: OptionsPanel

Reference to the OptionsPanel class which manages the game settings.

memberDetailPanel: GameObject

UI component that displays the detailed stats of the selected party member.

Methods:

SetState(UIState state):

Sets the currentState to the specified state. Calls the corresponding panel's open method and closes all other panels.

OpenPanel(UIState panelToOpen):

Based on the panel enum value provided, opens the respective panel.
For instance, if UIState.PARTY is passed, it would open the partyPanel.

CloseAllPanels():

Closes all individual panels. This can be useful for ensuring that no two panels are accidentally open at the same time.

TogglePanel(UIState panelToToggle):

If the provided panel is currently open, closes it. If it's closed, opens it.

UpdateCurrentPanel():

Calls the update or refresh method on the current panel, if applicable.

SwitchToNextPanel():

Based on the currentState, switches to the next panel in sequence.

SwitchToPreviousPanel():

Based on the currentState, switches to the previous panel in sequence.

Class:PartyPanel

Description: The PartyPanel serves as the primary interface for the player to manage their party composition. Through this panel, players can view each party member's statistics, switch out party members, and alter the main character's elemental type. A sleek UI shows portraits and vital stats, enabling players to make informed decisions about their group's configuration. The flexibility of this panel allows players to optimize their party based on upcoming challenges, ensuring they're always prepared for any encounter.

Attributes:

partyMemberIcons: List<GameObject>
    List of UI elements (icons/thumbnails) representing each party member.
selectedMemberIndex: int
    Index of the currently selected or highlighted party member.
mainCharacterElementIcon: GameObject
    UI element representing the main character's currently assigned elemental type (FireType).
healingItemCount: int
    Represents the current quantity of healing items available to the party. This count is shared among all party members.

Methods:

HighlightPartyMember(int index):
    Sets the visual highlight on the party member icon specified by the index.
    Updates the member detail panel to show the stats and details of the selected member.
CyclePartyMembers(int direction):
    Changes the highlighted party member based on the direction (e.g., +1 for next, -1 for previous). This method wraps around if at the start or end of the list.
UpdateMemberDetailPanel(PlayerCharacter member):
    Fetches details from the provided PlayerCharacter data and updates the displayed stats and other relevant details on the member detail panel.
Open():
    Activates the partyPanel, making it visible.Displays all party member icons and defaults to showing the first member's details (or previously selected member's details).
Close():
    Deactivated or hides the partyPanel.
SelectMember(int index):
    Directly selects a party member based on the index provided without cycling.
    Useful for scenarios like mouse clicks on specific icons.
ChangePartyMember(PlayerCharacter toRemove, PlayerCharacter toAdd):
    Swaps out a current party member for a new one.
    Updates the displayed party member icons and details accordingly.
ChangeMainCharacterElement(FireType newElement):
    Updates the main character's assigned elemental type.
    Refreshes the mainCharacterElementIcon to reflect the change.
DisplayAvailableFireTypes():
    Shows a list or selection UI of available FireTypes.
    Allows the player to pick a new FireType for the main character.
SelectFireType(FireType selectedType):
    Assigns the selected FireType to the main character.
    Updates any UI elements that showcase the main character's elemental type.

UseHealingItem(PlayerCharacter target):
Applies the healing effect to the targeted party member.
Reduces the healingItemCount by one.
DisplayHealingItemCount():
Visually displays the current count of healing items in a dedicated UI section, so players are always aware of their remaining stock.
IncrementHealingItemCount(int amount):
Increases the healingItemCount by the specified amount, typically used when the player acquires more healing items.
DecrementHealingItemCount(int amount):
Decreases the healingItemCount by the specified amount, primarily used when healing items are consumed.

Class:GearPanel

Description: players can view currently equipped items on their characters, see item descriptions, and swap gear. Each gear piece can influence a character's stats, so players can also immediately observe the effects of equipping different items. Additionally, with a system of locking, we can unlock slots over time

Attributes:

gearSlots: List<GameObject>
List of UI elements representing each gear slot.
selectedGearIndex: int
Index of the currently selected or highlighted gear slot.
gearDetailPanel: GameObject
UI component that displays details of the gear equipped in the selected slot.
availableGearList: GameObject
UI element that shows a list or grid of gear available to be equipped in the selected slot.

Methods

HighlightGearSlot(int index):

Sets the visual highlight on the gear slot specified by the index.
Updates the available gear list based on the slot type and character's available gear.
Open():
Activates the gearPanel, showing the gear slots for the selected character.
Defaults to showing the first gear slot's details.
Close():
Deactivates or hides the gearPanel.
SelectGearSlot(int index):
Directly selects a gear slot based on the index provided. Useful for scenarios like mouse clicks on specific slots.

DisplayAvailableGearForSlot(int index):
Lists the gear items that can be equipped in the selected slot, allowing the player to choose one to equip.
EquipGearToSlot(GearItem gear):
Equips the selected gear item to the currently highlighted slot. Updates the character's stats accordingly and refreshes the member detail panel. Player can't equip item if they already have gear on of that statType equipped
UnequipGearFromSlot(int index):
Removes the gear item from the selected slot.
Updates the character's stats accordingly and refreshes the member detail panel.
LockGearSlot(int index):
Locks the specified gear slot, preventing changes to its equipped item.
UnlockGearSlot(int index):
Unlocks the specified gear slot, allowing changes to its equipped item.
NOTE: I've decided the spell panel isn't' explicitly necessary for gameplay so low priority

Class:SpellPanel

Description: The SpellPanel serves as the hub for spell management within the RPG menu. Players can swiftly view, select, and rearrange spells for their characters,

Attributes:

currentSpells: List<Spell>
The list of spells currently equipped/active for the selected member.
availableSpells: List<Spell>
The list of spells available to the selected member that can be equipped or swapped in.
spellSlots: List<SpellSlot>
Represents the UI elements for each slot where a spell can be equipped.
selectedSpellSlotIndex: int
Index of the currently selected or highlighted spell slot.
mainCharacterStoredSpells: Dictionary<FireType, List<Spell>>
Stores the main character's spell list for each elemental type they've previously been.

Methods:

HighlightSpellSlot(int index):
Sets the visual highlight on the spell slot specified by the index.
Updates the available spell list based on the selected member's available spells and current type (if main character).
Open(PlayerCharacter member):
Activates the spellPanel, showing the spell slots for the provided member.
Defaults to showing the first spell slot's details.
Close():
Deactivates or hides the spellPanel.
SelectSpellSlot(int index):

Directly selects a spell slot based on the index. Used for direct interactions, like mouse clicks.

DisplayAvailableSpellsForSlot(int index):

Lists the spells that can be equipped in the selected slot, allowing the player to choose one to equip.

EquipSpellToSlot(Spell spell):

Equips the selected spell to the currently highlighted slot.
Updates the character's spell list and refreshes the display.

UnequipSpellFromSlot(int index):

Removes the spell from the selected slot.
Updates the character's spell list accordingly.

StoreMainCharacterSpells():

When the main character switches their fire type, the current spell configuration is stored in mainCharacterStoredSpells.

RetrieveMainCharacterSpells(FireType type):

Retrieves and sets the previously stored spell configuration for the provided fire type for the main character.

Temp:ItemsPanel

Currently temporary might have use cases for the narrative

Temp:OptionsPanel

Also currently temporary

Items

Class:CloneSystem:

Description: Manages the creation and management of clones, unlocking fire types, and managing clone behaviors.

Attributes:

unlockedFireTypes: List<FireType>

A list of fire types the player has unlocked.

activeClones: List<PlayerCharacter>

The clones that are currently active in the party or battle.

Methods:

UnlockFireType(FireType type):

Unlocks a specific fire type, allowing the player to use its magic and create clones of that type.

CreateClone(FireType type):
Generates a clone of the specified fire type, setting its base stats and growths based on the fire type.
RemoveClone(CloneCharacter clone):
Removes a specific clone from the active list.

ScriptableObject:HealingItem

Description: The HealingItem class defines the properties and behaviors of healing items available to the player. These items offer a set amount of healing and can be replenished at specific locations like campfires.

Attributes:

itemName: string
The name of the healing item, e.g., "Healing Potion".
description: string
A brief description of what it does.
healAmount: int
The amount of HP this item restores when used.
maxCount: int
The maximum amount of this item the player can carry at any given time.
currentCount: int
The current amount of this item in the player's possession.
icon: Sprite
A visual representation of the healing item for UI purposes.

Methods:

Use():
Applies the healing effect by restoring the healAmount to a character's HP.
Reduces the currentCount by one.
Add(int amount):
Increases the currentCount by the specified amount, ensuring it does not exceed the maxCount.
Reduce(int amount):
Decreases the currentCount by the specified amount, ensuring it does not drop below zero.
GetRemaining():
Returns the currentCount, indicating how many of this item the player currently has.

ScriptableObject: Gear

Attributes:

gearID: string
    A unique identifier for each gear piece. This can aid in lookup, especially if you're saving or loading game data.
gearName: string
    The display name of the gear.
gearDescription: string
    A short description or flavor text for the gear.
statType: statType
    Enum denoting the type of stat.
boostAmount: float
    A float containing the increase in stats
isPercentage: bool
    A float that determines if its a percentage increase
gearIcon: Sprite
    A visual icon representation of the gear for UI purposes.
isEquipped: bool
    A flag to check if this gear piece is currently equipped by a character.

Enum: FireType

Description: Represents the different types of fire (or elements) that can be associated with spells, characters, and potentially other game mechanics.

Values:

STORM: color-red
LIGHTNING: color-purple
SUN: color-yellow
Note: the ones below are tentative
RAIN: color-blue
CLOUD: color-green
SNOW: color-white
SKY: color-orange
None: no type

ScriptableObject:Spell:

Description: Represents a magical ability or spell in the game. Contains information about its effects, mana cost, and other relevant properties.

Attributes:

name: string
The name of the spell
manaCost: int
The amount of mana required to cast the spell.
basePower: int
A value representing the base strength or effectiveness of the spell. This can be used in damage calculations, healing amounts, etc.
unlockLevel: int
Level that user has to be to use this spell
numHits: int
Number hits applied for any given attack
spellType: FireType
Defines the fire type of this spell.
effectType: EffectType
The type of effect the spell has (e.g., damage, heal, status ailment).
description: string
A brief description or lore about the spell, which can be shown in the player's spellbook or UI.
spellAnimation: Animation
The animation or visual effect to play when the spell is cast.

Methods:

Cast(CharacterBase caster, CharacterBase target):
Executes the spell's effects from the caster to the target. This can be about reducing the caster's mana, applying the spell's effects to the target, playing animations, etc.

Nested Enums/Classes:

EffectType
Enum values might include: HEAL,AOE_HEAL, ADJACENT_AOE_DAMAGE, SINGLE_TARGET_DAMAGE, MULTIHIT_TARGET_DAMAGE, AOE_DAMAGE,BUFF,DEBUFF,SHIELD,BUFF_AND_MANA_RESTORE,AOE_BUFF

Utilities

ProgressManager

Description: The ProgressManager is dedicated to tracking the player's progress throughout the game. This includes keeping track of completed quests, unlocked areas, storyline progression, and any other progression metrics relevant to the game.

Attributes:

areasUnlocked: List or Set
    manage which game areas or zones the player has unlocked.
storylineProgress: Variable or structure
    track how far the player has progressed in the main story or any subplots.

Methods:

UnlockArea(Area area):
    Unlocks a new area for the player.
UpdateStoryProgress(StorySegment segment):
    Updates the player's progress in the game's story.

Class:SaveLoad

Description: Responsible for the serialization and deserialization of the SaveState, handling file operations to save game data to storage and to load it back.

Attributes:

saveFilePath: string
    The path where the save files are stored. This can be a directory path, and individual save files can be timestamped or named uniquely.
currentSaveState: SaveState
    A cached instance of the most recent or loaded save state, allowing for quick saves or retrievals without needing to read/write to storage immediately.

Methods:

SaveGame(SaveState state):
    Serializes the provided SaveState into a format suitable for storage (e.g., JSON, binary) and writes it to the save file path. Updates currentSaveState with the provided state.
LoadGame(string fileName) -> SaveState:
    Reads the specified save file from the save file path, deserializes it into a SaveState, and updates currentSaveState with the loaded state. Returns the loaded SaveState.
CreateSaveStateFromCurrentGame() -> SaveState:
    Interfaces with the GameManager or other relevant systems to pull current game data and create a SaveState.
Note: stuff below is extra don't worry about it very low priority
AutoSave():
    Uses the current game state to generate a new SaveState and saves it, potentially with a timestamp or specific naming convention to differentiate it from manual saves.
DeleteSave(string fileName):
    Removes a specified save file from the save file path, effectively deleting that save game.

GetAllSaveFiles() -> List<string>:
Retrieves a list of all save files present in the save file path, useful for presenting players with a list of their saved games.

Class:AudioManager

Description: Manages the game's audio, ensuring the correct playback of sound effects, music, and other audio cues. It can handle volume control, audio transitions, looping, and other related functionalities.

Attributes:

musicSource: AudioSource
Dedicated audio source for playing background music tracks.
sfxSource: AudioSource
Audio source used for playing sound effects. Multiple may be used if overlapping SFX are expected.
currentTrack: AudioClip
The audio clip currently playing as the game's background music.
volumeLevels: Dictionary<AudioType, float>
Keeps track of volume levels for different audio types (e.g., Music, SFX).

Methods:

PlaySFX(AudioClip clip, float volumeModifier = 1.0f):
Plays a specific sound effect. The volumeModifier can be used to adjust the volume for this specific playback.
PlayMusic(AudioClip track, bool loop = true):
Plays a background music track. If loop is set to true, the track will loop indefinitely.
StopMusic():
Stops the currently playing background music.
PauseMusic():
Pauses the currently playing background music.
ResumeMusic():
Resumes playing the paused background music.
SetVolume(AudioType type, float volume):
Sets the volume for a specific audio type (e.g., Music or SFX).
FadeOutMusic(float duration):
Gradually reduces the volume of the currently playing music over the specified duration until it stops.
FadeInMusic(AudioClip newTrack, float duration, bool loop = true):
Gradually increases the volume of a new music track over the specified duration.
MuteAll():
Mutes all audio.
UnmuteAll():
Unmutes all audio.

Nested Enums/Classes:

AudioType:
Enum values might include: MUSIC, SFX, etc.

ScriptableObject: EncounterManager

Description: Manages the spawning and respawning of enemy encounters throughout the game world.

Attributes:

encounterList: List<Encounter>
List of all possible encounters in the game.
defeatedEncounters: Dictionary<int, float>
Dictionary to store the timestamps of when specific encounters were defeated. The key is the encounterID and the value is the time of defeat.
respawnTime: float
General time in minutes/hours before an encounter can respawn after being defeated.

Methods:

EncounterDefeated(Encounter encounter):
Marks an encounter as defeated and logs the current game time.

CheckRespawn(Encounter encounter):
Checks if an encounter is eligible for respawn based on the elapsed time since it was defeated and the respawn conditions.
RespawnEncounter(Encounter encounter):
Respawns a specific encounter at its original location.

Nested Classes:

ScriptableObject:Encounter
Description: Represents a specific enemy encounter in the game world.

Attributes:

encounterID: int
A unique identifier for each encounter.
location: Vector3
The world coordinates of where the encounter is (or should be) located.
enemyGroup: List<EnemyCharacter>
The group of enemies that form this encounter.

Methods:

Spawn():

Instantiates the encounter at its specific location.

Defeat():

Marks the encounter as defeated and hides or disables it from the game world.

ScriptableObject: GameDataDatabase

Description:A centralized class to manage various game data types such as gear, spells, and dialogues.

Attributes:

gearList: List<GearSO>

References to all Gear ScriptableObjects.

spellsList: List<SpellSO>

References to all Spell ScriptableObjects.

statusEffectList: List<StatusEffectSO>

References to all Status Effect ScriptableObjects.

Note: might not need dialogue to be stored here, need to do some more research on scriptable objects

dialoguesList: List<DialogueSO>

References to all Dialogue ScriptableObjects.

Methods:

GetGearByID(int id):

Returns a specific Gear based on its ID.

GetSpellByName(string name):

Returns a specific Spell by its name.

GetStatusEffectByName(string name):

Retrieves a status effect scriptable object based on its name.

GetDialogueByID(int id):

Returns a dialogue sequence based on its ID.

Note: I really hope we don't need this class below, if we do it's either due to indoor areas or because I'm struggling with level design, so leave it alone for now

Class: SceneTransitionManager

Description: Manages all scene transitions, maintaining continuity and game state during switches between game scenes.

Attributes:

currentScene: Scene
    The active game scene.
nextScene: Scene
    The scene to transition to.
transitionEffect: Animator
    The visual effect used for transitions
loadingUI: CanvasGroup
    UI elements that appear during loading (e.g., loading bar, hints, lore snippets).
transitionDuration: float
    Duration of the transition effect.
audioManager: AudioManager
    Reference to the AudioManager for handling audio transitions.

Methods:

InitiateTransition(string sceneName):
    Initiates a transition to the specified scene.
LoadNextScene():
    synchronously loads the next scene while displaying the loading UI.
OnSceneLoadComplete():
    Called when the next scene finishes loading. Initiates the transition effect and swaps scenes when ready.
ApplyTransitionEffect():
    Executes the chosen visual transition effect.
SaveCurrentState():
    Captures and saves the current game state, such as player position, stats, and other relevant data.
LoadSavedState():
    Restores the game's state from saved data upon entering a new scene.
UpdateLoadingUI(float progress):
    Updates the loading UI with the current loading progress. Can also cycle through hints or lore snippets.

HandleAudioTransition():
    Manages fading out of current scene audio and fading in of next scene audio.

Class: TransitionTrigger

Description: In-game object that, when interacted with, initiates a scene transition.

Attributes:

triggerCollider: Collider
    The collider that detects interaction with the player or other entities.
targetSceneName: string

The name of the scene to transition to when this trigger is activated.

transitionManager: SceneTransitionManager

Reference to the SceneTransitionManager to handle the actual scene switch.

Methods:

OnTriggerEnter(Collider other):

Checks if the player has entered the trigger zone and initiates the scene transition. (Use