Tower of Druaga Project Progress Record

Character Data

|  |  |  |  |
| --- | --- | --- | --- |
|  | Character |  | Class |
| 0 | Death | 0 | Hp |
| 1 | CurHP | 1 | S1 |
| 2 | MaxHp | 2 | S2 |
| 3 | CurS1 | 3 | S3 |
| 4 | MaxS1 | 4 | S4 |
| 5 | CurS2 | 5 | Str |
| 6 | MaxS2 | 6 | Vit |
| 7 | CurS3 | 7 | Int |
| 8 | MaxS3 | 8 | Hit |
| 9 | CurS4 | 9 | Spd |
| 10 | MaxS4 | 10 | Act |
| 11 | StrMod | 11 | Red |
| 12 | VitMod | 12 | Orange |
| 13 | IntMod | 13 | Yellow |
| 14 | DexMod | 14 | Green |
| 15 | AgiMod | 15 | Blue |
| 16 | ActMod | 16 | Purple |
| 17 | HpMod | 17 | AggroGet |
| 18 | EquipHead |
| 19 | EquipBody |
| 20 | EquipLHand |
| 21 | EquipRHand |
| 22 | EquipFeet |
| 23 | EquipAcc1 |
| 24 | EquipAcc2 |
| 25 | White |
| 26 | Blue |
| 27 | Black |
| 28 | Red |
| 29 | Green |
| 30 | AggroGet |

The way I’ll handle character data.

For each character, create a Character object, which will store only their equips, any alterations to their stats that are permanent for a particular play-through, their current stats which are a result of their base numbers in addition to any alterations for easier access, which will be recalculated at any appropriate time, it can also be changed on the fly with buffs and debuffs but can be restored to it’s intended value by referencing the stats from the specific Job Class class of the character, along with any particular permanent alterations that play through.

There will be a class separate by itself with a list of all skills usable by player characters in the game, each being a method of it’s own.

The method would take in only one variable, being the caster of the skill. The skill methods will run, and if the skill requires targets, it will ask the player for input. Regardless of required target input, it will end with a confirmation saying, “cast this spell? (Y/N)

The jobclassesinfobox class will have member functions for each of the different classes that will take in an int parameter for the level of the character.

The functions will have a case loop for all the levels ranging from 1 to 30, and for each case will return a nsarray that contains all the base values of a character in that class at that level.

It will also have a second function that takes in a text label of a class, and returns a nsarray of text labels of all skills learnable by this class.

The physicalcharacter class will have a member function that takes in a nsarray and assigns all the values to that characters parameters accordingly, as well as then apply and character specific modifiers. It will have member functions that retun it’s class, as well as all of it’s different stats and values.

Items

I’m going to have 3 ItemsBox classes that will contain functons for every item in the game.

Then there will be a physicalItem class that will be an object that represents a single particular item.

There are 3 main types of items: Consumables, Loots, and Equipments. There will be one ItemBox class for each, so ItemBoxConsumables, ItemBoxLoots, and ItemBoxEquips.

Loot:

There is one function that takes in a string of the item in question, and the function will return a nsarray based on case for that string input of the information of the item.

|  |  |
| --- | --- |
| Disc | Flavor Text and discripton of the item. |
| White | Value in particular element of the item. |
| Blue | Value in particular element of the item. |
| Black | Value in particular element of the item. |
| Red | Value in particular element of the item. |
| Green | Value in particular element of the item. |

Consumables:

There are two functions, both take in a text label of the equip item, one that will return an array of the properties of the item, and one that will run an action and do something specific to that item, ie, heal. Array is same as loot array.

Equips:

There are two functions, both takes in a text label of the equip item, one that will return an array like the other two, with the description containing the stats of the item, and one that will retun a nsarray of stat modifiers.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Atk | Def | Str | Vit | Int | Hit | Spd | Hp | S1 | S2 |
| 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 |
| S3 | S4 | White | Blue | Black | Red | Green |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 |

The 5 different colors will each align itself to one of the 6 modifiable stats.

White modifies DEX

Blue modifies INT

Black modifies AGI

Red modifies STR

Green modifies VIT

How am I going to implement the skills?

There will be a class, SkillsBox that will have a function that will take in a number that represents a skill, and then call another function in the class appropriately based on that skill.

There will be another function in the class that will solely return the description text of a skill based on a passed in number that represents the skill.

The way loot will be used in this game.

All characters will start off with predetermined equipment.

Each piece of equipment in the game however is unique, however more than one character can start off with the same piece of equipment, ie, knight and defender both starting with the same “Iron Plated Armor”

All pieces of equipment will give ATK and DEF values, and sometimes-special traits of effects as well.

Implementation wise, equipment do not internally store special effects, but rather, when a character gets a new piece of equipment that has a special effect, ie “attract more agro” or “increase act” or “increase spell lvl2 casts”, it is worked into the character him or herself when he obtains it and stays there permanently.

This basically means no starting pieces of equipment actually has special effects, even if it says it does, for example, “Guardian Ring” Saying it gives the wearer increased agro or “Mind Ring” saying it gives the wearer increased INT and S1 casts does NOT ACTUALLY do that, but rather, that class will automatically start with those stats at lvl1.

Characters can never unequip or switch equipment; they can only upgrade them by combining loot onto their equipment.

All pieces of equipment have threshold values for evolving into a new piece of equipment. When you combine a piece of loot with a piece of equipment, it will add a specific amount of Color Affinity to the character wearing that equipment based on the loot, ie “Red Gem” will give the character +8 Red Affinity.

However, it will also add to that particular equipment’s internal Affinity Value, which does not affect the character in any way, it only keeps track of the amount of affinity so far imbuned to it. When it reaches one of it’s several threshold values, ie “Red+10, Black+7”, the equipment will “Evolve” into a new piece of equipment, with different ATK and DEF values, and may or may not apply a new special effect to the character. Again, this special effect is not directly worked into the equipment, but when the character gains the equipment, it is directly added to the character’s Modifier fields. It is structurally the same as the character consuming a stat modifying item.