Goals

- Make a MUD

```
Info          Material          Enchantments
   \             |             /
    \            ↓            /
     →         Item         ←
          /    |    \    \
         /     |     \    \
   Weapon  Clothing  Consumable  Landmark  Container
```
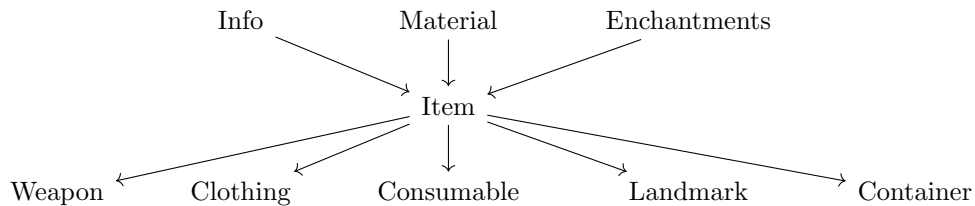
Descriptions

Item: Has data members durability, weight, weight-distribution, flexibility,

Forward Declared Dependencies

1. **Info:** Contains external description of item. This includes how it looks, size, and volume-type. Size and volume type factor into what said item can fit into and weight distribution. Also contains unique ID and name.

2. **Material:** Contains information regarding the material of the item. This factors primarily into things like durability, weight, density, and flexibility.

3. **Enhancements:** Contains information regarding how this item affects the statistics of the one who posesses it or the one it's used on.

Child Classes (Dependants in which 'Item' is Included)

1. **Weapon:** Has data members damage-type.

2. **Clothing:** Has data members defense, wearable-locations.

Possible Commands

Weaponry

jab, light swing, medium swing, heavy swing, parry, guard, special

Clothing

wear

Ideas

- Combat like Vindictus/PSO. It needs to be *that* good.

    - Step forward when swing (using attacks as movement)
    - Fast-paced

- Environment/tiles/depth like MineCraft

- Graphics like Final Fantasy Tactics (maybe taller)

- Card Game like FF8 or FF9

- A primary tower-dungeon like Sword Art Online (but with a whole world built around it; it doesn't consist of just the tower-dungeon).

- Play some .Hack and see how that combat is.

- No one can see each other's level, equipment, etc. explicitely (only how it looks, how strong a person is, etc.)

- NEURO: Concentrate for special attack. Black and white. Soothing voice: "Breath..." Everything around silences.

Game Engine Logic

Attack

Soul::attack() $\xrightarrow{\ 1\ }$ world::process() $\xleftarrow[2]{}$ Soul::defend()

abilities and buffs

1. Soul has member 'target' and member object 'stats'. References to these members are sent to the world, along with desired ability. Target is const reference to target.

2. World checks, first, to see if the ability is in Soul's ability list. If so, it applies ability's damage modifiers.

3. World calculates 'pure damage' using soul1 stuff. It calculates "pure defense"