Design - Epic Knight Adventure

Entity

+ Entity(std::string name, int baseAttack, int baseHp, int baseMana)
+ void fight(Entity *target, Types::attackType type)
+ int applyDamage(int value)
+ int calcDamage(Types::attackType type, int intensity)
+int getHealth()
+int getAttack()
+int getMana()
+std::string getName()
+ virtual Types::factionType getFaction() = 0
+ virtual ~Entity()

int baseAttack;
int baseHp;
int baseMana;
int attack;
int hp;
int mana;
std::string name;

Types

+ enum attackType{ PHYSICAL ,RANGED, MAGIC };
+ enum factionType{ PLAYER, NPC, MONSTER }
+enum actionType { MOVE_NORTH, MOVE_EAST, MOVE_SOUTH, MOVE_WEST, FIGHT }

Player

+ Player(string name):entity(name, 100, 20, 100)()
- vector<Item> inventory

+void addToInventory(const Item& item)
+void removeFromInventory(vecto<Item>::iterator it);
+const vector<Item>& getInventory()const;
+ void applyPowerUp(const Item& item)
+void displayInventory()
+void useItem(int index)
+void setAttack(int attack) { this -> attack = attack}
+void setHealth(int health) { this -> health = health}

Items

- int dmg;
- int hp;
- std::string name

Item(std::string itemName, int itemDmg, int itemHp)
std::string getName()
virtual ~item()

NPC

+NPC(string name, int baseStrength, int baseHp, int baseMana)
+NPC(string name)
+virtual void talk() cont
+Types::factionType getFaction()

Place

- Item item;
- NPC monster;
+

+ Place(const string &itemName, int itemDmg, int itempHp, onst string &npcNmae, int npcDmg, int npcHp, int npcMana);
+ virtual void printRoom Description();
+ void fightMonster(Player &player);
+void processChoice(int choice, Player &player);
+virtual string getSettingName() const = 0;
+ item getConstructedItem()
+NPC getConstructedNPC()

UI

+ field: type

+UI(){}
+ void printMap(int currPosition);
+char promptChar(std::string msg, std::vector<char> valid);
+Types::actionTYpe promptAction(std::vector<Tyes::actionTYpe>)

Director

+ Player *player;
+ Place *settings;
+ int position;
+ UI ui;
- Place * makeRandomPlace (int itemDmg, int itemHp, int npcDmg, int npcHp, int npcMana);

+ Director(Player *player)
+ ~Director()

DreamSymphony

FrozenTundra

KittyIsland

RizzlerCanyons
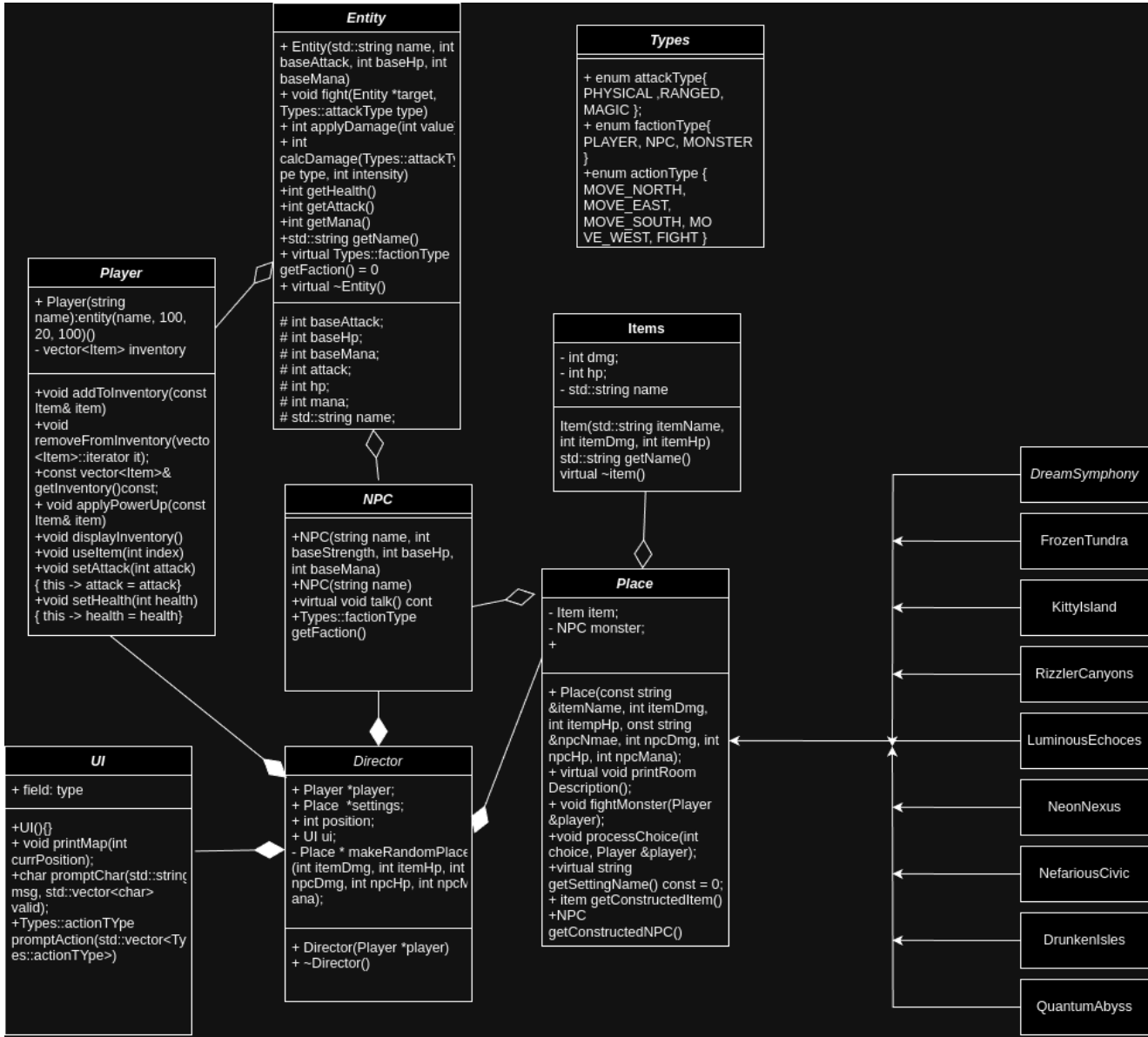
LuminousEchoces

NeonNexus

NefariousCivic

DrunkenIsles

QuantumAbyss

Class: Director
- Attributes:
  - player: Pointer to a Player object.
  - settings[9]: Array of pointers to Place objects.
  - position: Integer representing the player's position.
  - ui: UI object.

- Methods:
    - Constructor:
        - Parameters: Pointer to a Player.
        - Description: Initializes with a player and generates unique places.
    - Destructor:
        - Description: Destroys the object, freeing memory for places.
    - makeRandomPlace:
        - Parameters: itemDmg, itemHp, npcDmg, npcHp, npcMana.
        - Returns: Pointer to a dynamic Place.
        - Description: Creates a random place based on attributes.
    - doTurn:
        - Description: Manages player turns, map display, room descriptions, and actions.
    - fightMonster:
        - Parameters: Pointer to a Player.
        - Description: Initiates a battle with a monster in the current place.

Class: Entity
- #includes "Types.h"
- Constructor constructs a name, an attack, a base hp, and a base mana, doesn't matter if it's a player or a monster they both call from entity.
- Void fight is a type pointer that takes in the targets and takes in an attack type. From there it checks what it is and then the different types of attack have a different way of attacking.it points to the target hp and base hp and calculates the amount of damage done.

- Applying damage basically concludes how much damage is done. Sets hp to max(0, hp - value)
- Then the calc damage just checks for the different ways to do damage.
- Set of getters for each and every stat the player has.

Class: Item
- Attributes:
    - name: String representing the item's name.
    - dmg: Integer representing the item's damage.
    - hp: Integer representing the item's health.\
- Methods:
Constructor:
    - Parameters: itemName, itemDmg, itemHp.
    - Description: Initializes the item with a name, damage, and health.
getName:
    - Returns: String representing the item's name.
    - Description: Retrieves the item's name.

getDmg:
- Returns: Integer representing the item's damage.
- Description: Retrieves the item's damage value.

getHp:
- Returns: Integer representing the item's health.
- Description: Retrieves the item's health value.

Destructor:
- Description: Destroys the item object.

Equality Operator (==):
- Parameters: Reference to another Item.
- Returns: Boolean indicating whether two items are equal.
- Description: Compares items based on their properties (name, damage, health).

Class Place:
- The Place class is an abstract base class representing locations in the game.

It contains an Item and an NPC as private members, encapsulating the concept of a place with interactive elements.

The class has a pure virtual method printRoomDescription() that must be implemented by derived classes.

Public methods like getItem() and getNPC() provide access to the encapsulated elements.

- Derived Place Classes:

Several derived classes (DreamSymphony, FrozenTundra, etc.) extend the Place class, representing specific places in the game world.

Each derived class has a concise constructor initializing the base class with specific properties for items and NPCs.

The printRoomDescription() method in each derived class provides a unique description for that place.

- Dynamic Polymorphism:

The use of an abstract base class (Place) enables dynamic polymorphism, allowing the creation of instances of different derived classes through a common interface.

- Header Guards:

Header guards (#ifndef, #define, #endif) are used to prevent multiple inclusion of header files, ensuring clean compilation.

- Encapsulation:

Encapsulation is employed by keeping the Item and NPC members private within the Place class, allowing controlled access to these components.

Class Monster: Creates an entity with a name, base strength, base hp, base mana.

Class NPC: Creates an entity with a name, base strength, base hp, base mana, and creates an entity with a couple of other additional functions. Void talk basically just lets the character say something, and get faction returns where they're from, say monster vs player faction.

Class NPC