CS246 Final Project CC3K UML Diagram
Rivers Chen, Leah Lin, Hanyu Xu

All pointers * in this UML diagram is implemented using shared pointer

Enemy and Treasure and Potion depends on Board as they are spawned by Board. Similarly, GameController has dependency with Architect and PC,. For simplicity of graph, we omit lines here.

GameController
- pc: PlayerCharacter *
- stairX: int
- stairY: int

+ resetFloor(in: istream&): void
+ readFloor(in: instream&): void
+ readGE(x: int, y: int &, c: char, race: char,
    sqr: vector<GameElement *>&): void
+ setLevel(lvl: int): void
+ spawnPC(c: char): void
+ setGameElement(): void
+ moveEnemy(): void
+ movePC(direc: string): void
+ drinkPotion(direc: string): void
+ attackEnemy(direc: string): void
+ displayLose(): void
+ displayWin(): void
+ displayScoreboard(): void
+ levelComplete(): bool
+ setPCChamber(int chamberInd): void
+ flushDisplay(): void
+ isEnded(): bool

ScoreBoard
- score_board: vector<Score *>

+ clear(): void
+ write(): void
+ read(): void
+ add(score: int, name: String): void
+ sort(): void
+ display(): void

Score
- score: int
- name: string

+ getScore(): int
+ getName(): string

Board
- floor : vector<vector<vector<GameElement*>>>
- enemyList: vector<String>

+ spawnPotion(): void
+ spawnOnePotion(x: Integer, y: Integer, prob: Integer): void
+ spawnGold(): void
+ spawnEnemy(): void
+ spawnOneEnemy(x: Integer, y: Integer, prob: Integer): void
+ moveEnemy(pc_x: Integer, pc_y: Integer): void
+ moveOneEnemy(e: EnemyCharacter*): void
+ setRow(newRow: vector<vector<GameElement*>>): void
+ revert(ge: GameElement*): void
+ revert(x: Integer, y: Integer): void
+ enemyDeath(e: EnemyCharacter&): void
+ replace(ge: GameElement*): void

<friend>

Display
+move(gev: vector<vector<vector<GameElement*>>>&): void
+ getNotified(pc: PlayerCharacter*): void
+ action(act: ACTION): void
+ action(act: ACTION, int): void
+ action(act: ACTION, damage: int, l: Living*,l: Living*): void
+ action(act: ACTION, p: Potion*): void
+ newAction(s: Sring, preceed: bool): void
+ clearAction(): void
+ interpAction(act: ACTION): String
+ flush(): void

**<Enumeration>
Action**

MoveEA
MoveWE
MoveNO
MoveSO
MoveNE
MoveNW
MoveSE
MoveSW
ReachStair
SeePotion
SeeGold
SeeEnemy
Win
Die
AtkEnemy
GotAtk
DrinkP
PickG

GameElement
+ display(): void
+ getNotified(pc: PlayerCharacter&): void = 0

Goblin
+ slain(race: char): void
+ attackedBy(orcs: Orcs&): void

Troll
+ incHp(): void

Drow
+ getAtk(): int
+ getDef(): int
+ attackedBy(elf: Elf&): void

Vampire

Shade

Ghost

PlayerCharacter
- score: Integer

+ getAtk(); Integer
+ getDef(): Integer
+ getScore(): Integer
+ notify(ge: GameElement*): void
+ getNotified(pc: PlayerCharacter&) void
+ drink(p: Potion*) void
+ pickup(p: Treasure*) void
+ resetPotion(): void
+ attack(c: EnemyCharacter&): void
+ attackedBy(human: Human&): void
+ attackedBy(dwarf: Dwarf&): void
+ attackedBy(elf: Elf&): void
+ attackedBy(orc: Orcs&): void
+ attackedBy(merchant: Merchant&): void
+ attackedBy(dragon: Dragon&): void
+ attackedBy(halfling: Halfling&): void

Living
+ getAtk(): Integer
+ getDef(): Integer
+ getNotified(PlayerCharacter &pc): void = 0

NonLiving
+ getNotified(pc: PlayerCharacter&): void = 0

Architect
+ getChamberInd(): Integer
+ getNotified(pc: PlayerCharacter &): void

EnemyCharacter
+ getNotified(pc: PlayerCharacter&): void
+ attack(pc: PlayerCharacter&): void
+ attackedBy(human: Human&): void
+ attackedBy(drow: Drow&): void
+ attackedBy(vampire: Vampire&): void
+ attackedBy(troll: Troll&): void
+ attackedBy(goblin: Goblin&): void
+ death(pc: PlayerCharacter&): void

Treasure
+ getNotified(pc: PlayerCharacter&): void
+ getAmount(): Integer

Potion
+ getNotified(pc: PlayerCharacter&): void
+ effect(pc: PlayerCharacter&): void

Potion decorates PC

Dragon Hoard
- dragon: Dragon*

+ getAmount(): Integer
+ getNotified(pc: PlayerCharacter&): void

Small

Normal

PotionDef
+ effect(pc: PlayerCharacter&): void

PotionHp
+ effect(pc: PlayerCharacter&): void

PotionAtk
+ effect(pc: PlayerCharacter&): void

Elf

Orcs

Dragon
+ attack(pc: PlayerChacracter*): void
+ attackedBy(pc: PlayerCharacter*): void
+ getNotified(pc: PlayerCharacter*): void

Halfling
+ attackedBy(pc: PlayerCharacter&): void

Human
+ death(pc: PlayerCharacter&): void

Dwarf
+ attackedBy(vampire: Vampire&): void

Merchant Hoard

creates upon death

Merchant
- hostile: bool

+ death(pc: PlayerCharacter&): void
+ getNotified(pc: PlayerCharacter&): void
+ attackedBy(human: Human&): void
+ attackedBy(drow: Drow&): void
+ attackedBy(vampire: Vampire&): void
+ attackedBy(troll: Troll&): void
+ attackedBy(goblin: Goblin&): void

PlayerCharacter;s attackBy takes in EnemyCharacters, and EnemyCharacter's attackedBy takes in PlayerCharacters, here, we show a dependency line between PC and EC, and for clarity of whole UML, we omit lines between subclasses.