Class and Package Diagram for System TwitterNethack

Assignment in the course PA1415 Software Design

2017-04-11

Authors and Authors contribution

Author Name	Social Security Number	Thinking	Writing
Max Tlatlik	19940813-XXXX	20%	-
Samuel Sarin Drysén	19950113-XXXX	15%	50%
Fredrik Olsson	19910114-XXXX	15%	-

System Description

The goal of this system is to create a game based on the idea of the game Nethack. The system shall adopt the rules of the original Nethack to keep its flare.

The most interesting feature for the system is the map generation where the maps can be generated with the twitter API. The map generation work in a way that the users can generate a map in at least two different ways.

The system also includes multiplayer support. User are able to access games online via the internet of offline via a LAN.

Overall Packages

List of all packages

- **Entities**
  Contains the classes regarding in-game entities such as:
  - ❏ Player character
  - ❏ Enemies
  - ❏ NPCs

- **User Interface**
  Contains the classes that build up the user interface such as the main menu and inventory screen in-game.

- **Instance**
  Contains the classes that handle dialogue and combat encounters.

- **Game World**
  Contains the classes for levels, level-generation, and items such as potions or food but also items that can be equipped like armor and weapons.

- **Layout**
  Contains the classes for laying out the game worlds map.

- **Network**
  Contains the classes that handle online connectivity for the multiplayer mode and twitter API connection for level generation.

- **Character Customization**
  Contains the classes that handle customization and player-class specific information
  such as:
  - ❏ Class
  - ❏ ClassDescription
  - ❏ ClassCatalog

- **Scene**
  Contains the classes for different game states such as in-Game and main menu.

Package Diagram

Character Customization

CharLobby

-attributeArr: int[0..*]

+makeNewCharacter(): void
+setAttribute(index, value): void
+finishCreation()

CharClass

ClassDescription

ClassCatalogue

+Create character

Game World

Layout

RoomManager

+enterRoom(pos, ID): void

-0..*

Room

-doorStatus: bool

+enterRoom(pos, ID): void
+recheckDoorStatus(): void

-0..*

Tile

Item

-pickUp: bool

+pickUpItem(): void

User Interface

ChatBox

+create(): void
+roomEnteredMsg(): bool
+doorClosedMsg(): bool
+intimidationMsg(): void
+playerWonMsg(): void
+playerLostMsg(): void
+fleeFailedMsg(): void
+fleeSuccededMsg(): void
+showPlayerMsg(theMsg): void
+showEntReply(reply): void
+itemAddedMsg(item): void
+failedToPickUpMsg(): void

-0..*

Inventory

+checkForKey(keyID): bool
+roomLeft(itemSize): bool
+addToInv(item): void

+1

Scene

Menu   Game

Instance

Battle

-combatStatus: bool
-entWeapon: Item
-entHP
-playerHP

+create(theEntity): void
+destroyEnt(): void
+combatAction(pItem)
+rollFlee(): bool

Dialogue

-leave: bool
-reply: string

+create(theEntity): bool
+sendMessage(theMsg): void

+Char uses User Interface

Entities

EntityManager

+startCombat(entID): void
+endCombat(theEntity): void
+startDialogue(entID): void

-0..*

Entity

-inCombat: bool
-inDialogue: bool
-weaponItem: Item

+startCombat(): void
+endCombat(): bool
+startDialogue(): bool
+endDialogue(): bool
+getEntWeapon(): Item
+attack(pItem): void
+getHP(): int

CharManager

+setAttribute(index, value): void
+makeCreation(attributeArr): bool
+enterRoom(pos, ID): bool
+attackPlayer(entWeapon): int
+markPlayerDead(): void

-0..*

Char

-attributes: int[0..*]
-weaponItem: Item

+create(attributeArr[]): void

+Network Connection

Network

Server   Client

+TwitterAPI level generation