Edward Hu, Oliver Dong, David Yang

***APCS Final Project:* ARPEGGIO**

**Abstract:**

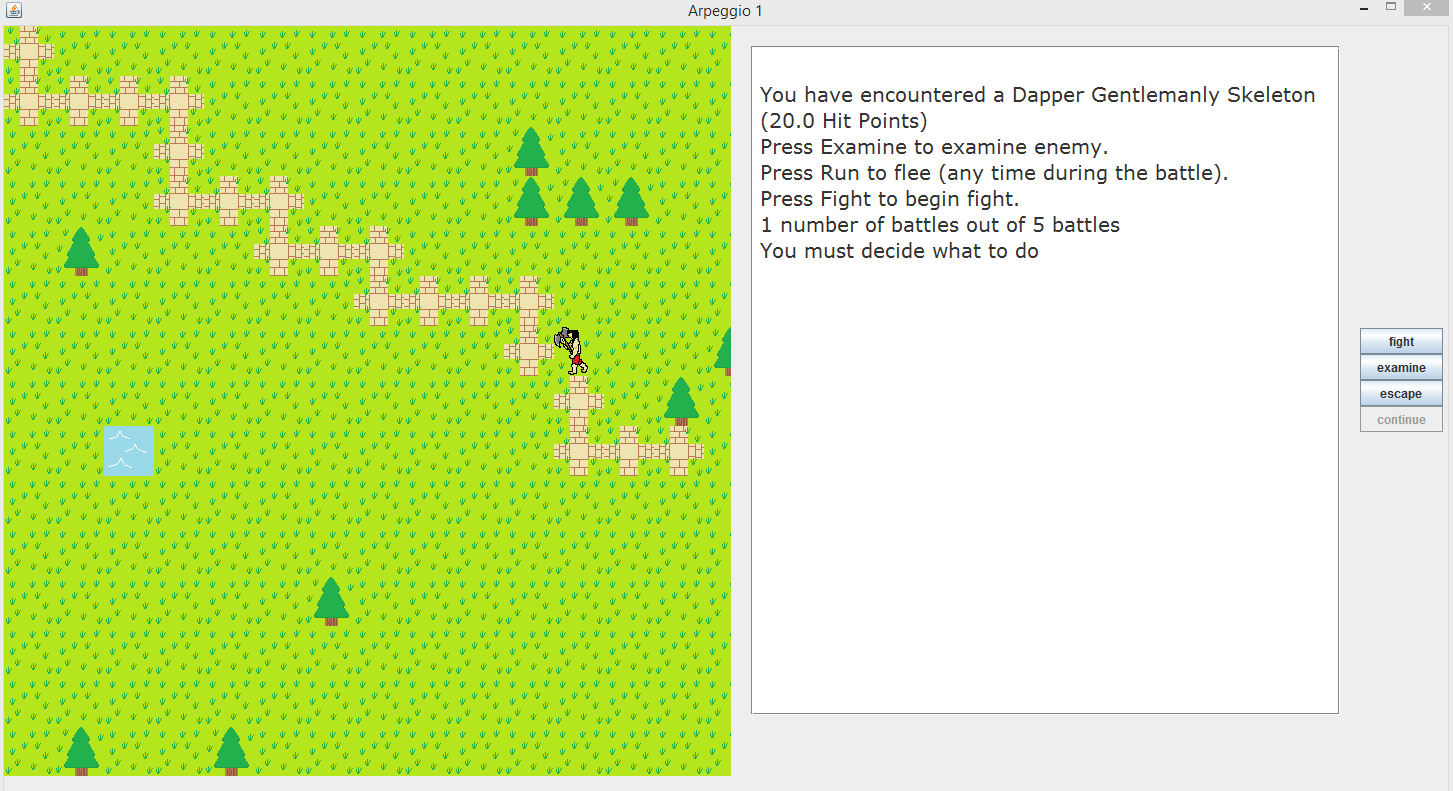
Arpeggio is our vision of a dungeon-diving Role Playing Game experience. A Role Playing Game (henceforth referred to as RPG) can be defined as a game where the participant assumes the role of a character, usually in a fantasy or science-fiction setting. The player can interact with the world around him/her. Usually, the player will fight enemies, acquire weapons and armor to augment their combat capabilities, and gain “experience points,” which reflect the veterancy of the player’s character. With enough experience points, a character can level up, increasing their “stats.” Stats are variables that represent natural combat prowess, such as agility, toughness, or muscular strength. The character will also gain money (sometimes referred to as “gold”) to purchase armor or weapons. Additionally, the player’s character has a set amount of “Health Points” or “Hit Points” (henceforth referred to as HP). When the character’s HP is reduced to zero, that character has been defeated/killed. In Arpeggio, our aim is to satirize common tropes found in RPGs (EX: tiresome mechanics, hackneyed storylines, use of convenient plot holes to advance a story) while at the same time, providing an entertaining, turnkey gaming experience.

**1: “Divide and Conquer”**

Starting the task,we decided to divide Arpeggio into two parts - the combat system and the GUI.

Arpeggio features a console turn-based combat system inspired by old-school classics such as Zork. The user can equip items from our vast library of weapons and armors, each possessing unique qualities which enhance the character’s combat capabilities. The inventory reads from a text file and implements a hashmap to store each item, enabling us to insert items without regard for item order. There are nine distinct “elements” in the game representing different kinds of harm, from fire to psychological. Additionally, players can choose between one of three “classes” of characters, with each class possessing different stats. For instance, a rogue might hit more often than a berserker, but when the berserker hits, he hits much harder.

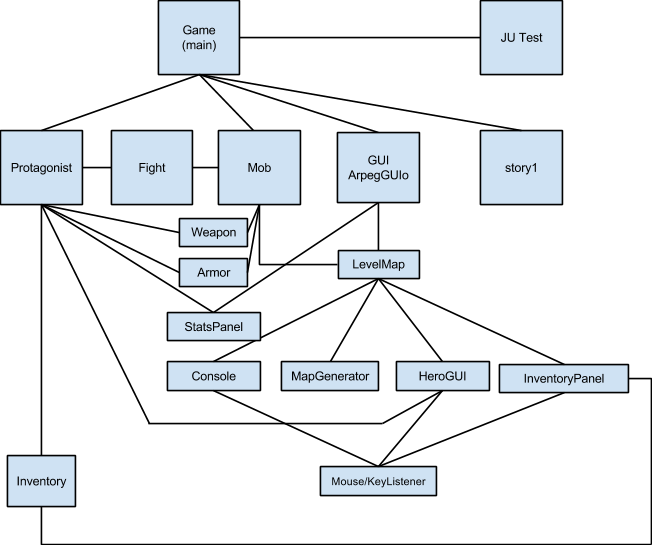
Arpeggio features a minimalist pixel-art based design and five randomly-generated maps to explore, helping to keep the game fresh and interesting on subsequent playthroughs. It uses AWT and Swing to create the front end graphics. The main components of the Window will be the main dungeon board, a console to display information about fights / quests, and an inventory pane on the side. The map generation will include roads, trees, and other elements, all randomly generated *(fig 1)*.



*fig 1: An example of the game overview screen.*

The player will use the keyboard and mouse to navigate himself through the world. He will eventually encounter monsters based on chance and tile type. The player will then be able to choose several options during the encounter, such as fight, examine, or run. During the fight, the player will be able to select weapons and armor for different situations. After the player levels up enough, the boss screen will appear and the user will fight an unique boss. After the boss is defeated, the user can collect loot and advance to the next level.

**2: Object-Oriented Design.**



**Game:** The main of the game. Initializes the weapon and armor hashtables, as well as the player and levels.

**Armor:** Has getter methods for armor element as well as percentage and static block. Contains a 6 digit stat that is manipulated for damage mitigation calculations.

**Weapon:** Has getter methods for weaponx element, hit chance, and damage. Contains a 6 digit integer which is manipulated for damage calculations.

**Protagonist:** Contains Armor, Weapon, Inventory, String Name, Stats, Character type, and experience points. Can level up after gaining sufficient experience points.

**FinalFight:** RNG, turn-based combat engine. Each weapon has 9 possible elements with varying effects. Manipulates Armor’s and Weapon’s 6-digit stat to emulate a battle. Damage mitigation from protagonist’s armor and the enemy mob’s armor is taken into account. Grants experience at end of battle. Level ups up the protagonist if sufficient experience is accrued.

**story1:** Reads text from file. Includes some ASCII art + story.

**ArpegGUIo:**  Holds all the JPanels together. It uses BorderLayout to put the console on the right, the map on the left and the inventory on the bottom. It takes in a level parameter to decide what kind of level to load in the map and what types of monsters the user will encounter.

**LevelMap:** A JPanel that depicts the Graphical map of the game. It uses MapGenerator algorithms to generate a map with roads, trees, etc. Also draws the Hero Tile onto the map using HeroGUI and listens for key presses in order to control the Hero. Every key press or player movement, depending on the level, will have a chance to start a fight with the enemy. Once the player reaches a certain level, the next encounter will be set to become a Boss Battle and a cutscene will occur.

**MapGenerator**: Creates a 2D array of integers. Randomly generates a lake, a set # of trees, and a road (which stops upon hitting a lake or a border).

* + Road involves 2D array recursion
  + Tree takes an integer for num trees given precondition: num trees < available tiles
  + Lake makes a rectangle of water tiles which are at most 1/5th the board’s length.

**HeroGUI:** Represents the Hero avatar in the game. Holds images for the hero, and also the hero directional images. There are different hero avatars / hero directional images based on class and level type. For example, heros will be swimming in the sea level and be in spaceships in the space level. Uses actionlistener to change hero’s directions based on key presses passed along through LevelMap.

**StatsPanel:** Takes in the stats from the masterProtagonist class and displays them on the GUI.

**InventoryPanel:** Uses JComboBoxes to display the current inventory of the protagonist, from weapons to armors. Uses listeners to select the correct armor and weapons and set them to the player’s current weapons and armor.

**Inventory:** Contains arrayLists of armor and weapons.

**Protagonist:** Contains the player’s stats and inventory. Used to pass info along to many classees such as FinalFight, StatPanel and InventoryPanel.

**Mob:** An object that models the enemies the player will fight. A list of Mobs is stored in ArpegGUIo because ArpegGUIo is level specific and we want level specific lists of Mobs. So space levels will have space mobs, and sea levels will have sea mobs. The mob information is passed into the FinalFight engine along with the player information.

**3: Detailed Design. Please reference our JavaDocs**

**4: Testing. Extensive beta testing phase / JUnit Tests. Test for various map sizes, game balance.**