Digging for Treasure

A text adventure inspired by the board game Incan Gold

Buy the board game here on Amazon

Here are the original rules

User Manual

Index:

Tim Hodson 4/26/17

Components:

Setting up the game is straightforward. There are 3 python files, an MP3 file, and a SQL database file.

Python Files: These 3 files must be stored in the same program folder.

1. dft2.py: This is the primary python file and runs the game.
2. db.py: This file creates a SQLite database and inserts 10 fake players with score 0. You are NOT required to run this file, as the submission already includes a SQL database. (See if you can beat one of the high scores!)
3. player.py: This file holds information for the player class used in dft2.py. You are not required to run this application, only save it in the same project folder.

MP3 File: This must be saved in the same program folder as the python files.

1. Ae2.mp3 This is the audio file that plays music for the game. You are required to install pygame if you want the game to play music, and you must also assign the music file's location on line 238 of dft2.py. Otherwise you can comment out lines 237-239, and 7 of dft2.py. This MP3 file is from the game "Ages of Empires 2" and I hold NO copyright or claims to the music.
   Age of Empires 2 The Age of Kings + The Conquerors OST. Composers: Stephen Rippy, Kevin McMullan. Link to go buy the game. I am not monetizing this project in any form, it is designed to purely be an open source parody of two great games.

SQL database File: This must also be stored in the same program folder as the python and MP3 files.

1. This is the SQL database created to store high scores. You are not required to use my database file. If you choose not to use mine you must run db.py to setup a new SQL database.

Setup:

1. First download the submitted files (3 python files, 1 MP3, 1 SQL DB file) and ensure they are saved together in a single project folder.
2. Ensure you have the required components installed (pygame, sqlite3)
    a. If you have difficulty installing pygame, you can comment out lines 7 and 237-239 of dft2.py. This will remove the soundtrack to the game.
    b. Otherwise, open dft2.py and navigate to line 238, and assign the file location of your MP3 file.
3. Run the dft2.py file to start the game.

Code Validation:

1. The validity of the code has been tested with multiple playthroughs both by myself and peers.
2. Single player, multiplayer, and AI components all work correctly as designed.
3. High scores save successfully in a SQLite database, checked with a SQL database browser.
4. All quest cards (treasure, traps, golden idols) all work correctly and their functions are described in the gameplay portion of this user manual.
5. I added comments to the code wherever I thought necessary to help in the comprehension of the code.

Gameplay:

1. To start a new game, run dft2.py. You will be greeted with a short introduction and high score list.
2. To begin, enter the number of human players who would like to play the game, I recommend 1-5 players. You will be then asked to enter the name of each player.
3. Next you will be asked if you would like to play with an AI opponent. Enter "y", "Y", "YES", or "yes" to add an AI player, or no if you do not want a computer opponent.
    a. If you choose to play with an AI opponent, you will then be asked if you would like to play with an easy or hard AI player. Enter "e", "E", "easy", "EASY", "Easy" for an easy opponent. Any other entry will default you to a hard opponent.
    b. The easy AI player's name is Simple Simon.
    c. The hard AI player's name is AI-Tron-4000.
4. Once all players have been added, game play will begin. You have 5 days to recover as much treasure as possible. Each day you will go through a series of quest cards. There are 3 unique types of cards you will encounter:
    a. Treasure Cards –15 in deck- Each treasure card you encounter will have a value between 1-20. The treasure amount displayed will be split evenly between any remaining players in the cave. Any remaining treasure that could not be evenly split players is added to the bounty. (See Golden Idol below for more information on the bounty)
    b. Golden Idols –5 in deck- A Golden Idol card adds 5 coins to the bounty. Think of the bounty as all the remaining treasure left behind that could not be picked up by players. The first time someone leaves the cave for the day, they can pick up the bounty on their way out. If more than one player leaves the cave at the same time, they evenly split the bounty and any remainder starts a new bounty. This creates a dilemma for the player whether to press their luck and venture farther into the cave, or call it good and grab some extra coins on the way out.
    c. Trap Cards –15 in deck- There are 5 different types of traps, and 3 copies of each trap in the deck. If you encounter 3 of the same trap before leaving the cave,

any remaining players lose their coins and must return to camp. The possible traps are Spiders, Rock Slides, Lava Floods, Spike Pits, and Snakes.

5. After each quest card is played, the remaining player's scores will be displayed. Then each player will take turns deciding if they want to keep venturing further into the cave for more treasure, or cut their losses and leave to camp. You do not want to tell other players if you are staying or leaving, as this can alter their own decision to go deeper into the cave or leave. This is why your text input is hidden during gameplay. Once a player leaves for camp, they can no longer collect coins for that day. But when a player leaves, they store any coins collected in their bank. After everyone has left the cave (or 3 of the same traps are encountered), the day ends and a new day begins.
6. At the end of the game (5 days) whoever has the most coins in their bank wins.

Conclusion:

This was an incredibly fun project to work on and I am extremely thankful I was given the opportunity to use it as my final project. I learned a lot about python from this class, and this was a great way to put my knowledge to use. It was really interesting to develop an algorithm for the AI opponents, and it actually works quite well. The hard opponent can be pretty difficult to beat! The AI algorithm is a similar concept to counting cards in blackjack. The computer determines the odds of getting a card that would cause failure, and if it's over a certain threshold (0.05 for easy AI, 0.10 for hard) then the computer player leaves the cave. It was also interesting to learn about SQL, and find out the fantastic integration python has with SQLite 3. This allowed me to capture high scores and query them in a 1980's arcade game fashion. I would eventually like to develop a GUI for the game to make it more interactive for players, so maybe that will be a project this summer. If you want to track the progress of this project or others like it, follow me on GitHub!

Tim Hodson

GIS Programming Project Proposal

4.10.17

My project proposal is to turn a board game, Incan Gold, into a computer run text game. This includes generating the "deck" of cards a group of players uses. I will detail the basic idea of the game below. The program I will write will uniquely generate the game for the players, include multiplayer support, and include a text narrative to immerse the players into the board game "world".

The game Incan Gold is a risk-taking game where players explore a cave in search of buried treasure. But the cave also has many traps and pitfalls the players can fall victim to. The game consists of 5 "days" that adventurers can explore the cave. As the adventures explore they collect treasure, run into traps, and possibly find golden idols. The idea of the game is to collect the most treasure by the end of the 5 days. As each quest card is flipped, you have the option to keep exploring deeper (risking losing all your treasure), or leaving for camp so you can store your treasure. If you leave though, you cannot collect any more treasure for that day. If you find a treasure card, it has a corresponding value between 1-20. The treasure is then split between the remaining players. If the treasure cannot be evenly divided, then any remainder is left in the cave. Whoever leaves the cave first picks up this extra treasure on their way out. If more than one player choose to leave at the same time, they split the remaining treasure. As players leave the cave, the reward for the remaining players increases (as does the risk). There are 3x of 5 different trap cards. If you venture far enough into the cave to encounter 3 of the same trap, then you lose all your currently held treasure and must retreat to camp. The interesting part of the game is that players do NOT tell each other if they are going to stay or leave the cave, so the entire game is based on the idea of pressing your luck.

It's a fantastic little card game that is a very fun party game. I believe it would make a fantastic and enjoyable python project. To be fully transparent, I started coding this game in python this past Saturday for fun. I have included my current code below. It is nowhere near complete right now, and only supports a single player.

Thank you,

Tim Hodson