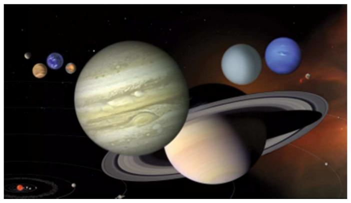
CMPT 120,
Fall 2016, SFU
Instructor: Diana Cukierman
Final assignment: "Planets, aliens and explosions" CMPT 120 game, ©Diana Cukierman

FINAL ASSIGNMENT: "Planets, aliens and explosions" game


NASA

Shutterstock

This is a ***<u>up to 3 (three) team people work</u>*** *(you can also work in pairs or individually if you prefer so). Consider carefully if you want to develop this assignment on your own since it involves considerably more work than previous assignments.*

You may discuss generalities with colleagues from other teams about this assignment, ***however you cannot develop the same code, nor share code or diagrams among teams, nor obtain code or diagrams from any other sources****.*

Being a team exercise it places a big responsibility on each individual. You want to respect your partner/s and yourself: DO YOUR SHARE, and BE KNOWLEADGEABLE OF THE WHOLE PROJECT. Keep in mind that exercises related and analogous to this project will likely be part of the final exam. ***Hence, as you are working on this assignment you are also preparing for the final exam.***

<u>Submission deadlines (2 DEADLINES):</u>

FIRST DEADLINE: <u>Thursday Dec 1, 11:59 pm</u>. (CodeWrite) *<u>Set of Codewrite exercises related to the assignment</u>. All team members should submit the exercises via Codewrite, even if they developed them together. You are encouraged to work on your own for further practice.*

SECOND DEADLINE: <u>Monday December 5, 11:59 pm.</u> (Canvas) *THE DEADLINE IS FIRM, it is the very last possibility for you to submit.* ***No exceptions can be made****. A <u>solution will be posted immediately after the deadline.</u> You are encouraged to submit before the deadline.* ***<u>You are highly encouraged to start working on this assignment as soon as possible.</u>***

Details of the files to be submitted are described below.

<u>A. PROBLEM SOLVING</u>

1. FIRST: Read the Problem Solving Suggestions document. IT IS BRIEF AND HIGHLY RECOMMENDED.

2. **READ THIS WHOLE DOCUMENT**. **UNDERSTAND WHAT YOU ARE ASKED TO DO** AND WHAT YOU ARE ASKED TO SUBMIT. FIRST READ THE SECTION TITLES AND THEN THE DETAILS. TAKE NOTES, DO DIAGRAMS, DISCUSS WITH YOUR TEAMMATES, ASK IF IN DOUBT.

B. GENERAL DESCRIPTION OF THE “PLANETS, ALIENS AND EXPLOSIONS” GAME.

[NOTE: to ease the description, the player/astronaut and the user are referred to as male buy clearly the player and user can be also a woman!]

3. You are asked to implement an adventure game. The game is played by **one player/astronaut** and there are “computer players”/aliens, emulated as part of the game. The astronaut travels through space, arriving to different **planets** as he travels. The player/astronaut starts the game with some **fuel** (measured in liters), and an empty room in his spaceship to collect **rock specimens**. He travels aiming to reach a special planet **“PythonPlanet”.** In his travels he will lose fuel and possibly be able to increase his fuel as well (details are described below)

4. The astronaut/player may encounter aliens in the planets (of possibly lesser, equal or greater **civilization level**). Civilization levels may be 0,1,2 or 3 . Depending on the respective civilization levels the player may be attacked or attack the aliens, affecting his fuel (as will be explained). He may end up stranded in some planet if he runs out of fuel (and considered dead in that case) .

5. “PythonPlanet” has an “infinite” source of rock specimens and fuel, and has no aliens. If the player arrives at PythonPlanet he stops the travelling as he has reached the best of all planets and wins (and the game is over). Unluckily, the player cannot know in advance which planet is PythonPlanet, but he keeps travelling hoping to have good luck; eventually he may (or may not) arrive there.[NOTE: for testing and for submitting , you are required to show the user which planet is PythonPlanet]

6. **Explosions** may happen. “Explosions” cause that more rock specimens are made available in neighbour planets as the game unfolds, as described below. Additionally , “Amazing explosions” (as opposed to “Mild explosions”) destroy the planet where the explosion occurred. See below.

7. The user will be able to play more than one game. With a new game a new board starts, and the player is back in planet #0. **When each game is over and also when all games are over some information** will have to be shown to the user. (details below)

B.1 How this world is represented

8. The world is a board, with many planets. Each planet has:
   a. Aliens of a certain civilization level (or an indication that there are no aliens , i.e. level 0).
   b. Fuel (in liters)
   c. A number of rock specimens available

9. The player /astronaut will have:
   a. A name (that the user will provide)
   b. A civilization level (0 to 3 that the user will provide)
   c. A current position in the board (starting in planet# 0)
   d. A number of fuel liters (that the user will provide)
   e. A room in his spaceship containing (a list of) rock specimens collected (empty at the start of the game)

10. The planets constitute the board game , and are represented by **several lists , each list representing some information about the planets.** (For example, one list will have the rock specimens in each planet, another list will contain the fuel liters in each planet, etc). A position across all lists will have the information associated to one planet (in that position).

11. **The planets will be named or numbered based on the position they occupy in the lists**.

12. ***The lists are circular**:* after visiting the last planet in the list the subsequent planet is the first one in the lists.

13. NOTE: A **text file** with planets information will be provided . **Python code to read such text file** into structures (lists) in your program ***is also provided***. More details and recommendations about the creation of the lists are described below.

C. THE GAME IN MORE DETAIL:

C.1 BEGINNING OF A GAME

14. The user will be asked to provide some initial information including which features to include in the game and initial data about the player. See details below and sample runs.

C.2 PLAYER MOVEMENT AND INTERACTION WITH THE USER

15. The player/astronaut starts his travelling in (and may come back to) planet #0. Planet #0 is a no-action, safe planet, where the fuel and rock specimens are not affected in any way and there are no aliens. When the player reaches this planet he just waits for his next turn. Planet #0 cannot have explosions and it cannot be PythonPlanet either.

16. The game unfolds by the player /astronaut **advancing in the board** to some other planets by **rolling one regular die (values 1 to 6)** (and counting circularly). The game unfolds until the player plays a certain number of turns, wins or gets stranded (because of running out of fuel). The maximum number turns is provided by the user at the start of the game. As the player moves explosions may happen.

17. **For each move or turn you will also have to provide the option that the player goes to a specific planet (as guided by the user)**. [NOTE: This is a **required** feature and will be highly useful for you to debug the program and to be marked]

18. Once a player **arrives to a normal planet** (not PythonPlanet) , **actions happen in the following order**: [NOTES: **Both the player and the planet may be affected to reflect the changes.** Only **integer values** are considered, truncate (consider the integer part only) of values as needed if obtaining fractional values].

a. If there are aliens in the planet, the player will interact with the aliens as follows:
   i. If the player has less civilization level that the aliens in the planet, the player will lose fuel, a random value, between 1 and all the fuel he has. (i.e. in this case he may lose all his fuel and thus get stranded in the planet and be considered dead). The planet fuel is not affected in this case.
   ii. If the player has equal civilization levels as the aliens, he may lose fuel , again a random number, but only from 1 to half as many fuel liters as he had so far (if the player's current fuel liters are only 1 he may lose that whole 1 liter, in which case he gets stranded). The planet fuel is not affected in this case.
   iii. If the player has higher civilization level than the aliens , or if there are no aliens in the planet (aliens civilization level =0), the player collects a random number of fuel liters, from 1 up to as many fuel liters as there are in the planet. The planet is affected to reflect that there is less fuel.

b. If the player does not get stranded in the planet, the player collects (the integer part of) 1/3 of the rock specimens in the planet. The planet is affected; it will have less rock specimens. The player/astronaut will keep the rocks in a different compartment in the room [This will be represented as a list, the new number of rocks will be appended to the list/ room] .

c. Then the player/astronaut will wait for the next turn in the game to continue travelling, unless he lost all his fuel in which case the player is considered to be stranded (and die).

19. If the player **arrives to PythonPlanet** he instantly wins.

C.3 "EXPLOSIONS"

20. There is the possibility that an "**explosion**" occurs. An explosion may happen randomly in some planet . **Explosions have to be checked before the astronaut moves**.

21. Any planet (except Planet#0) may explode. The calculation is done so that the planet position number may be outside the game board, in which case no explosion is considered to affect the game (i.e. one can think of it as if the explosion occurred in another galaxy)

22. The planet position is randomly calculated as a value between 1 and the length of the board multiplied by 5, hence allowing approximately 1/5 possibility to have an explosion within the board each turn (and 4/5 outside the board). No explosions are attempted if the player just won or just got/stranded.[NOTE: you can change the frequency of explosions when debugging]

23. **EFFECT OF EXPLOSIONS**: There are two levels of explosions: MILD and AMAZING.

a. **MILD EXPLOSION**: Intuitively, a mild explosion in one planet causes more rock specimens be available in the planet where the explosion takes place and also in neighbour planets , in a sort of cascade effect. All planets remain in the board.

b. More specifically: If a **mild explosion** takes place in a certain position (planet) within the board limits (but not in planet #0) (let's name the position ***posExp***) ,then some planets in the board get more rock specimens as follows: Planets that are in positions ***1*** and up to but not including the position *posSExp* (where the mild explosion happens) will have additional rock specimens; for any planet in position K, 1≤ K < posExp, planet K will be added the rock specimens that are available in the higher positions in the board (K+1, K+2 …) up to and including the position *posExp*. Notice that the calculation should be done starting from planet 1, gradually up to posExp , treating the list normally (i.e. not circularly).

i. For example if posExp is 3, and planets in positions 1,2, and 3 respectively have 10,20 and 30 rock specimens, then after the mild explosion takes place , the planets will respectively have 10+20+30, 20+30, and 30, that is, 60, 50 and 30 rock specimens
ii. Another example: if posExp is 3, and planets 1,2 and 3 respectively have 10,10 and 10 rock specimens, then after the mild explosion takes place , the planets will respectively have 10+10+10, 10+10, and 10, that is, 30, 20 and 10 rock specimens [NOTE: check the sample runs]

c. **AMAZING EXPLOSION.** [This level is for assignment bonus points] The effect of this kind of explosion is the same as the mild explosion as far as the neighbour planets concern. However, **the planet where the explosion occurs disappears from the board**. **That is, the board shrinks!** If the astronaut was in that planet, he dies (and the game is over). If the astronaut was in another planet the only thing that may happen to the player is that his current position may change (to adapt to the smaller board). The planet numbers (of the remaining planets) would get changed but this is determined by the planets positions in the lists, with no need to explicitly change the planets information. See sample runs.

C.4 END OF THE GAME – Winning results

24. The game finishes when the player has played the maximum number of turns or when he dies (i.e. gets stranded in a planet (loses all his fuel) or dies in an amazing explosion), or when he wins (i.e. reaches the special planet PhytonPlanet).

25. Once the game is over the user is shown final results including winning results and information associated to the player . The user is then asked if he/she would like to play again. [NOTE: See sample runs].

C.5 END OF ALL GAMES, INFORMATION SHOWN TO THE USER, CONCLUDING CALCULATIONS

26. You should show final information to the user analogous to what is shown in sample runs: if the player won or lost, rock specimens collected, etc. See the sample runs.

27. Additionally, **at the end of all the games (when the user does not want to play any more)** concluding calculations will be shown to the user. You will have to calculate a number as follows:

a. Consider the list of rock specimens left in the planets in the last game played. Based on that list create a new list with 0's and 1's so that in each position of this new list there will be a 0 if the corresponding number in the list with rock specimens in the planet is even, and it will be 1 if the number of rock specimens is odd.

b. Then consider the list with 0's and 1's to be a binary number, where position 0 in the list is the most significative bit.

c. Calculate the number in base 10 that the list of bits represents. For example, if there are 5 planets in the board at the end of the last game, and the list of remaining rock specimens in the planets is [10,5,3,4,8] then the associated binary list should be [0,1,1,0,0] (representing the number $01100_{(2}$) and thus the final number will be 12.

<u>D. DIALOG WITH THE USER, SOWING RESULTS, INITIALIZATIONS</u>

28. The dialog with the user will be done via text messages (i.e. with a text user interface) and optionally (for bonus points) the user will be shown a graphical display of the board using turtles.

29. The following Information will be <u>asked from the user before starting each game</u> :

    a. <u>The name of the file containing the planets information</u>. One file will be provided for testing: "planetsData1.txt". You will be able to create more files with different information for you to test different board situations (following the exact same format)

    b. <u>number of maximum turns</u> that the game will play in the game

    c. The user will have the option to indicate whether Explosions will take place or not and which kind of explosions and the proportion of explosions to be produced. [NOTE: It is highly recommended that you do not implement Explosions until you have the core working. Also, you are recommended to implement mild explosions first and in a subsequent step, amazing explosions]

    d. For each player the user is asked:
        i. The player's name
        ii. The player's civilization level
        iii. Each player initial fuel liters

30. **Information to be <u>shown to the user</u> (to keep the user posted about the evolution of the game):** [NOTE: See the sample runs]

    e. Each time after the player moves and at the end of the game it should be shown:
        i. the updated board, with the information associated to each planet: number of rock specimens available, the civilization level of the aliens in the planet (if any) , the number of fuel liters available
        ii. the planet should also indicate if it is the special planet PythonPlanet.

iii. the player's state including: name, current position, current fuel liters, current rock specimens, whether he is alive or not (i.e. if the player dead or stranded, the game will be over) .

iv. any message indicating events as they happen – such as the player fighting with aliens, explosions, etc

f. Additionally, as an optional feature (providing bonus points), after each player moves and also at the end, the board will be graphically drawn with turtles functions. The graphical drawing can distinguish if the planets have fuel left or none, the number of rock specimens, the special planet PythonPlanet may be marked also, it can also be marked where the player is, where explosions take place, etc. You can color code. You may want to leave the graphical details to the end; only add more details to the graphic part when your problem is solved.[NOTE: See the sample runs, captured screens and demo]

E. Other materials provided to you additionally to this description

31. A generic description document (1 page) about problem solving
32. Several sample runs, including captured screens and the text dialog shown to the user.
33. A piece of Python code to read a list of planets from a file into a list of strings (each string in the list will have the data for one planet with a specific coding)
34. A text file with planets information (which will be possible to be read by the provided code). You may create other files adding/revising planets to the file to further test your game, following the same format.

F. REQUIREMENTS: FEATURES, DESIGN AND DOCUMENTATION

35. Organize this game in a **modular** way, creating functions as necessary. This will give you more points when marked, and it will make your programming task feasible and much more easily manageable.
    a. **Come up with a top level description of the game and gradually work on the details**
    b. Think of useful functions to implement to deal with the player, with the board, etc.
    c. Functions proposed to solve with Codewrite (course #32) should help you as well. You will be allowed to use posted solutions from the Codewrite questions.

36. **The values asked to the user should be validated.** [NOTE: See sample runs]

37. Plan ahead which **variables and which data structures you will have** (your data structures will be essentially lists) you will have to define to keep all the data for easier access. HINTS: You will likely want to have:
    a. variables for general information (e.g. total games played, turns played in a game, etc.)
    b. variables for the player information (e.g. current position, points, civilization level, etc.)
    c. several lists to keep the information associated to the planets. The elements in the different lists with the same index would correspond to the same planet. (e.g. the p^{th} element in the rock specimens list would be the rock specimens in the p^{th} planet , the p^{th} element in the civilization level list would be the civilization level of aliens in the p^{th} planet, etc.)
    d. flags indicating some states (the astronaut won, etc..)

e. Decide a **clear naming convention of variables and functions and stick to it**. [NOTE: This is good practice and it will aid you and the marker]. You will deal with many variables and functions. Name your variables so that the name reminds you its role and purpose. For example, variables related with the astronaut may start with “a”, such as aFuel, etc.

f. In your functions you may use **variables via passage of parameters and return and/or also** you may include some variables (such as representing the board or the player) as **global variables**.

38. Include **comments in your functions** describing the input parameters, the output (i.e. returned) values , any global variables that may be checked and/or changed in the function and what the main purpose of the function is.[NOTE: This is good practice and will aid you and the marker]

39. You are **HIGHLY RECOMMENDED TO IMPLEMENT AND TEST this game in stages**, for example,
    a. create a top level core of the program and call (or invoke) functions which as a first stage may not yet be implemented but just have a “Tracing” print – so that you can run the program and get a general idea (given those printed messages) in which order the functions are called
    b. Gradually implement functions. Test them independently, then call (or invoke) them and combine them together (and continue gradually)
    c. Start by implementing the basic characteristics. Leave “explosions” for later, etc.
    d. See the various sample runs, they will suggest a possible order of implementation of the various stages

40. The graphics may allow you a better visual but they are optional.

G. ADDITIONAL ASSUMPTIONS

41. Additional assumptions may be made as long as they do not alter this description of the application. Any additional assumption should be justified and be well documented. If in doubt, ask the teaching staff.

H. FILES YOU NEED TO SUBMT BY DEADLINE II (Final submission):

42. **Files to submit**
    a. The code (Python file) of your final submission,
    b. Flowchart of your top level program
    c. At least 3 Sample runs in text files with some different situations (with an explanation of what is happening at the top) similar to what was provided with the problem description,
    d. Captured screens of the associated turtle graphics (if you implemented such)
    e. An admin text file explaining how you got organized with your team members, and any special situations. Name your admin text file **individually.txt** if you worked so. Any additional comments may be added in this file.

I. IF NEEDED, ANY FURTHER CLARIFICATIONS WILL BE ANNOUNCED IN CLASS AND IN CANVAS

End of Final Assignment description