Interplanetary Artillery Game documentation

This document is a documentation on the 2D game „Interplanetary Artillery Game“ which was written in Python using Pygame (and NumPy) by Alimhan Musalatov.

The player controls a tank and plays against a simple artificial intelligence on 4 different planets where each planet has its own gravity and terrain.

General Aspects

How to play

The program is started by running the file *main.py.*

- **Start screen** When starting the game / program, the user is presented with a start screen in which they can choose the planet to play on, using the arrow keys UP and DOWN and confirming the choice with ENTER.

- **Main game** Having entered which planet to play on, the main game starts. The player controls a tank by using the arrow keys LEFT and RIGHT to move the tank on the ground (which depends on the chosen planet), UP and DOWN to adjust the firing angle and SPACE to fire missiles. The goal is to get 3 points, one point is earned by hitting the computer enemy 2 times.

- **End screen** If the player or the AI enemy reaches 3 points, an end screen appears, displaying the score, the winner and options to play again or end the game. As in the start screen, the options are chosen with the keys UP and DOWN and confirmed with ENTER.

Game Settings

- **Planets** The planets to choose from are Earth, Moon, Mars and Ice Planet (fictional, probably).
- **Gravity** The gravity varies based on selected planet.
- **Missile velocity** The initial velocity varies based on the selected planet (to account for the different gravities).
- **Ground terrain** The ground on which the tanks move are generated based on mathematical functions, each different for the different planets.

Technical Aspects

Game components

- **Tanks** Tanks are modeled by a class called *tanks* that has methods for shooting, moving, angle adjustment, falling and reloading. The attributes of the tank are also realized by the class attributes of the class including for example position, (firing) angle, current moving direction, health / life, number of available missiles and frame which corresponds to the firing angle. The attribute frame dictates which imagine of the tank is shown (each imagine showing a different firing angle). Another important attribute of the tank class is the attribute *missiles.* This attribute is a list that contains all missiles that are fired from the tank and have not hit the ground, the other tank or that are not out of the window yet.

- **Missiles** Missiles, that can be fired by tanks using the method *shoot*, are modeled by a class called *missile.* The main attributes of the class describe the current position of the missile and the initial velocity vector (that depends on the angle it was shot from). With a specified time step, the method *position_update* calculates the next position of the missile depending on the current position and the initial velocity vector.

- **Ground** The ground is modeled by a matrix that is also called *ground*. The matrix has the same dimensions as the game window whose entries are either 0 or 1 so that each entry of the matrix represents a pixel. If the value of entry is 1, it means there is ground at the corresponding pixel, otherwise there is not ground. Initially, all the entries of the matrix are initialized as zero. Afterwards, the actual ground matrix is generated using a ground function (depending on the planet). The ground function takes in an x-coordinate (column of the ground matrix) as input and returns the corresponding y-coordinate (row of the ground matrix) representing the height of the ground at that point. After that, all entries below the y-coordinate in the given column are set to 1. The ground matrix is then used to draw the ground using the function *pygame.draw.rect* for each column. If a missile hits the ground the shape of the ground is changed by setting the entries within a certain radius from the point of impact of the ground matrix to zero.

- **AI enemy** The computer enemy is modeled by a class called *AI_enemy.* It is responsible for making decisions regarding movement, shooting, reloading and adjusting the firing angle. Its import attributes are the distance of the last missile that was fired from the player to the computer tank and the time the computer tank last shot. The methods describe the decisions the AI enemy can make. These are the ones listed above which are mostly random but the method *decision_running.* This method leads to the tank running away when the attribute distance gets too small (a missile from the player hit near the computer tank).

Main Functions

The main functions are given by the following:

- ***start_screen()*** Opens a starting screen on which the player can choose a planet to play on.
- **artillery_game(planet)** Main function which depends on the chose planet in the starting screen. It initializes the game environment, creates player and AI tanks, sets up game window and starts the game loop.
- **end_screen(score, winner, planet)** Opens an end screen when the player or the computer reaches 3 points. It shows the winner, the score and provides options for the player to choose from.

All three main functions utilize pygame for the corresponding infinite loop to show and update the screen.

Images

The game utilizes various images loaded from folders that exist in the same folder where the python files *main.py* and *tank.py* exist.

The folder t*anks_imgs* contains the different images of the tanks that are used in the game. There are 6 images for each tank, each image with a different firing angle. The images were drawn in Notability using an iPad. They are loaded by the function *load_images* and treated as attributes of the instances of the tank class.

The folder *backgrounds* contains different background images for the different possible planets and a background image for the starting screen. The background images are taken from the site pixabay.com which provides free images.