Tanks Game

Explanation of the game:

Having the *Pocket Tanks* game as a base, in the game of Tanks, players control tanks that are placed on opposite sides of a 2D battlefield. The objective is to use missiles to attack and destroy the other player's tank. With each turn, players can either move their tank a certain distance in any direction or adjust the power and angle of their next missile shot. The power determines how far the missile will travel, and the angle determines its trajectory. Once the power and angle are chosen, the player must fire the missile by pressing a button. The missile will travel in a parabolic path, potentially hitting the other player's tank and causing damage. If a tank sustains enough damage, it will be destroyed and the game will end.

Game architecture:

The game is organized into several files according to object-oriented programming and an images folder that stores every asset used as images for this game. There is a total of 9 python files for the game development: colors, terrain, bullet, tank, sprites, fsm, input_handler, multicast_transceiver, and finally main file.

There are several color constants in the RGB color model in the **colors** file, where each color is defined as a tuple of three integers representing the color's red, green, and blue values, with values ranging from 0 to 255.

The **Terrain** class represents a terrain object in a game, which has attributes for the width, height, and vertical position of the terrain in pixels, as well as a friction coefficient that is currently not implemented, but could be used to slow or speed up the tank's movement or event change the bullet's behavior, like making it reflect or pierce through the terrain.

The class **Bullet** has a method shoot which resets the bullet's parameters to be ready for shooting, activated when a player hits the shoot button and a method update which updates the bullet's position according to its angle, power, and gravity while the bullet is in the air, following the equations of movement.

The **Tank** class has properties such as name, health, position, and controls, as well as a finite state machine and an input handler. It has methods for moving and updating the tank's actions based on input events, a method for shooting the bullet according to the angle and power defined, and calculates the damage received from an enemy bullet based on distance.

A **sprites** file is a collection of classes that handle various aspects of a tank game, including creating and updating game objects sprites (tanks, bullets, terrain), allowing for assets to be loaded into the game, and collisions between the object to be detectable, using pygame.Rect objects. It also allows the updating and rendering of objects attributes like health and name of tanks as well as the image of tanks according to the current percentage of health it has.

The **fsm** file defines classes used to control the behavior of the tanks. The State class represents a state in a finite state machine (FSM) and the Transition class represents a transition between two states. The FSM class represents the FSM with a list of possible states and a dictionary of possible transitions between states. There are 5 possible states for the

FSM, IDLE, MOVING, FIRING, RELOADING, and DESTROYED. The IDLE state is the initial and standby state where the tank is not moving and waiting for the other tank to finish its move, the MOVING state is where the tanks can move or change the angle and the power of the upcoming missile, the FIRING state is active when the tank fires a missile, the RELOADING state stays until the bullet either exit the screen or hits any surface, and the DESTROYED state is when the tank's health drops bellow 0, meaning it was destroyed by the other tank.

The **input_handler** file defines classes and methods for handling user input in a game. The Command class is an abstract base class with an execute method that must be implemented by subclasses for specific commands. The InputHandler class maps user input to Command objects via a dictionary and has methods to get the controls for both tanks, according to what is requested. The ANGLE_LEFT, ANGLE_RIGHT, POWER_DOWN, POWER_UP, MOVE_LEFT, MOVE_RIGHT, and SHOOT classes are subclasses of Command that perform the movement, aiming, and shooting actions on a Tank object.

The **multicast_transceiver** file contains functions for sending and receiving multicast messages through UDP sockets. This file creates sockets that listen for multicast messages on a specified group and port or sockets for sending multicast messages while also allowing sending of a message through a given socket to a specific multicast group and port.

The main file holds the game logic, where every function, class, and object referred to is implemented.

Firstly, the tanks, bullets, terrain objects, and sprites are created and adjusted to the game dimensions while also initializing the game variables. Next, the input of the user is stored, collisions are detected between the bullets and the surfaces (tanks and terrain) and outer limits, the current player decision is implemented and finally, tanks are updated. Lastly, we have the rendering part, where first is rendered the background, next is rendered all sprites (bullets, terrain, tanks), next is drawn the preview line for the tank's missile, as well as its health and names. In this function, it is also drawn an explosion animation with sound effects that last 1 second when the missile of a tank hit any surface. Lastly, it was implemented a networking connection between players to play.

There are 3 total screens, 1 is the game itself presented previously and the second is the end screen, where the winner is presented until either the game is closed or the game is reset. The last screen is presented in LAN mode when one player is waiting for another to enter the game.

To start the game, simply execute the main.py python file. It will prompt you to enter a username and a mode, either local (0) or LAN (1). The user ID will be automatically generated when you begin the game, allowing you to play in LAN mode.

Patterns used:

The game is implemented using object-oriented programming and inheritance, with elements such as tanks, bullets, and terrain represented as objects and the pygame library is used to handle graphics and sound.

The code utilizes the command pattern to process user inputs, with the InputHandler class serving as a filter and the Command class and its child classes serving as an abstract base for specific command subclasses.

The prototype pattern is used in tank generation, with the Tank class having a clone method that creates a new tank with the same attributes as the original.

The state pattern is employed to manage the actions of tanks, with the State class and its child classes representing states and a FSM class that updates the tank's state based on specific events handled by the tank.

The game uses a double buffer to prevent flickering, which is implemented using the flip method from Pygame and the main game loop follows the game loop pattern, comprising of processing user input, updating game elements, and rendering the game to the display.

The update pattern is used to update the positions of tanks and bullets using sprites, with the TankSprite and BulletSprite classes having an update method that is called by the sprite group.

The subclass sandbox pattern is used in the sprites and states, with the BaseSprite class serving as a base class for TankSprite and BulletSprite and the State and Command classes serving as a base class for its various subclasses.

Innovative aspects:

The game implements collision detection in two ways: by using the collision of rectangles between bullets and tanks and by calculating the intersection point between the lines of the bullet's trajectory and the terrain. If the lines intersect, it is determined that a collision has occurred, and the point of collision is saved to represent the collision on the surface of the terrain. The location at which the bullet hits is used to calculate the damage to the enemy tank.

Additionally, the game has animation and sound effect that plays when a bullet hits any surface (tank or terrain). The bullet hit animation plays for a certain number of frames, and the explosion sound is played only once per animation.

The game's networking feature allows two players to play against each other over the internet using multicast communication and a sender and receiver socket. There are two modes of play: local and LAN. In local mode, players control their tanks using two sets of controls on the same keyboard. In LAN mode, four types of messages are used for communication between players: WAITING, READY, SYNC, and DATA. Players send WAITING messages to each other while waiting for additional players to join. When another player joins, they send a WAITING message to the waiting player to indicate that they are also waiting. Both players then send READY messages to each other, where the player with the lower ID randomly determines which player will start as the first tank. The final SYNC message is used to confirm that both players are synchronized, and the DATA message is used to transmit the actions of one player to the other player in real time.

Future Bug Fixes:

1. Any player can control both tanks by sending key press input to the LAN.

2. Each tank can move freely across the entire map without incurring a cost, but a fuel parameter must be implemented to track their movement. The fuel would restore each turn but would be drained as the tank moves.

3. The tanks can destroy each other by simply touching each other, as the bullet is stored within the tank's body and there is no obstacle preventing the tanks from overlapping.

4. The networking may not function correctly outside of the computer, possibly due to latency and message-sending limits.

Code link:

https://github.com/CarlosValente93092/TankGame

References:

- The game is based on the original "Pocket Tanks" game: https://classic.blitwise.com/ptanks.html
    - The game mechanics and overall design were inspired by the original "Pocket Tanks" game.
- Spritesheet: https://www.deviantart.com/cosbydaf/art/Tank-spritesheet-587230376
    - The spritesheet was used to create the graphics for the tanks and bullets in the game.
- Explosion sound: https://www.shockwave-sound.com/sound-effects/explosion-sounds/Explosion2.wav
    - The explosion sound was used when a missile hits any surface in the game.
- Dirt: https://www.freepik.com/free-photos-vectors/dirt-texture
    - The dirt texture was used to create the terrain for the game.
- Grass: https://www.freepik.com/free-photos-vectors/grass-texture
    - The grass texture was used to create the terrain for the game.
- Explosion: https://www.pngegg.com/en/search?q=explosion
    - The explosion image was used to create the visual effect when a missile hits any surface in the game.
- Background: https://www.vecteezy.com/vector-art/7079444-blue-sky-background-with-clouds
    - The background image was used to create the sky in the game.