Project Documentation For M11 Vehicles and DB

Author: Mason Fraser
Date: 08-Nov-19

Subject

Table of Contents

1. Project Overview

Implement JDBC Database into existing 3D Vehicles, save their state in a database so it can be reloaded from previous state.

2. Project Requirements

1) Complete outstanding Vehicles updates from the midterm, using proper OOD with an abstract (base) class.
2) Use a database to store data (How many Vehicles , of which type, where, and how fast...likely one table, enough to see that the start continues from the last ending). The Vehicles at the start come from the database, from the completed run of last time the program ended.

3) Use a sequence diagram to show relevant movement of data in the system.

3. Database Implementation

JDBC has been implemented to allow for a save state, the system will save on a button press (Numpad_7), which places all the items on screen into an array and send it of in a insert query to the database, on next boot, everything saved on screen will return in its last-saved position.

3.1. Persistent Database

Used a simple persistent database application provided by Karl Meissner.

3.1.1. Saving the Data

```
4.      for (GameItem items: gameItems) {

            double posX = items.getPosition().x;
            double posY = items.getPosition().y;
            double posZ = items.getPosition().z;

            try {
                Thread.sleep(100);
                if (items instanceof LandVehicle) {
                    db.update(String.format("INSERT INTO vehicles(veh_type,
    posX, posY, posZ) VALUES('LandVehicle', %f, %f, %f)", posX, posY, posZ));
                }
                if (items instanceof Motorcycle) {
                    db.update(String.format("INSERT INTO vehicles(veh_type,
    posX, posY, posZ) VALUES('Motorcycle', %f, %f, %f)", posX, posY, posZ));

                }
                if (items instanceof AirVehicle) {
                    db.update(String.format("INSERT INTO vehicles(veh_type,
    posX, posY, posZ) VALUES('AirVehicle', %f, %f, %f)", posX, posY, posZ));
                }
            } catch (Exception e) {
                e.printStackTrace();
            }

        }
        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
```

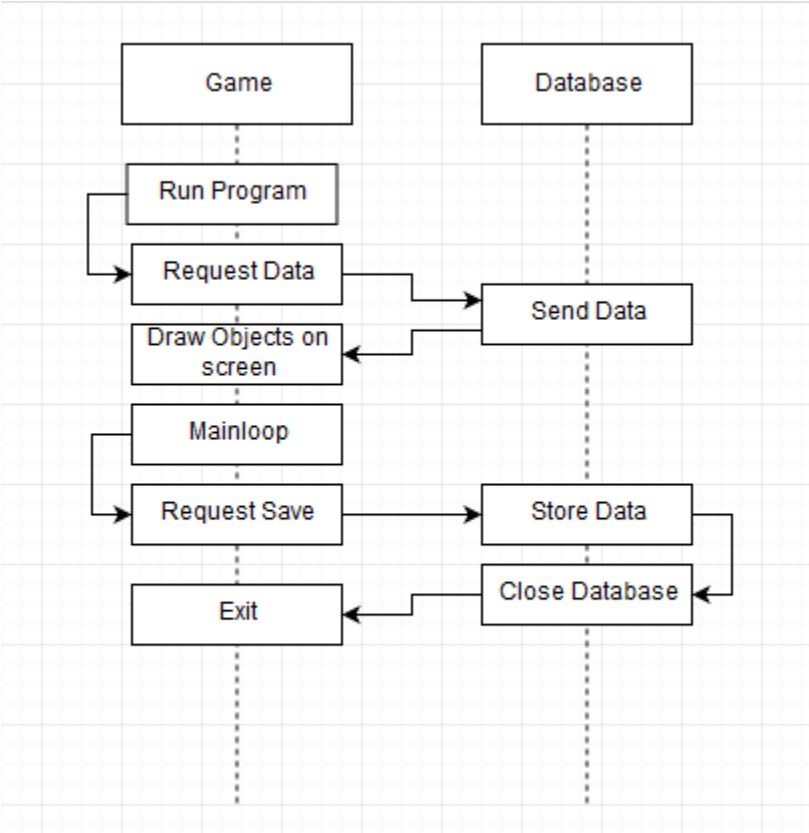
Here we are checking what type of vehicle something is and adding it into the database with its conditions. Ex. XYZ location, and the exact type of vehicle it is, to differentiate between each upon loading.

4.1.1. Loading Data

```
5.  Statement st = DriverManager.getConnection("jdbc:hsqldb:file_db", "sa",
    "").createStatement();
    ResultSet rs = st.executeQuery("SELECT * FROM VEHICLES");
    scene.setGameItems(new GameItem[]{terrain});
    for (; rs.next();) {
        Vector3f vector = new Vector3f(rs.getFloat(3), rs.getFloat(4),
    rs.getFloat(5));
        if (rs.getString(2).equals("LandVehicle")) {
            Vehicle house = new LandVehicle(vector, 0, 0, 100);
            house.setVelocity(0.002f, 0.000f, 0.003f);
            house.setRotationVel(new Quaternionf(0.06f, 0.01f, 0.03f, 0.0f));
            house.setScale(0.150f);
            scene.setGameItems(new GameItem[] {house});
        }
        if (rs.getString(2).equals("AirVehicle")) {
            Vehicle house = new AirVehicle(vector, 0, 100);
            house.setVelocity(0.01f, 0.01f, 0.01f);
            house.setRotationVel(new Quaternionf(0.06f, 0.01f, 0.03f, 0.0f));
            scene.setGameItems(new GameItem[] {house});
        }
```

Here we handle checking what is in the database, and creating new objects based on what they are and adding them to the game scene, you can see I use rs.getFloat() and the index of what value is stored in the database's internal array, we iterate to check if there is another object in the database, if there is…. Add it!

5.1. Sequence Diagram



This diagram shows the sequence of data being passed around the database and game client.