МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____ Литвиненко О.Є.
«_____»_____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ
"МАГІСТР"

Тема: «Геоінформаційна система аналізу і візуалізації даних трекінгу джерел поповнення енергії»

Виконавець: Ященко К. А.

Керівник: Халімон Н. Ф.

Нормоконтролер: Тупота Є. В.

Київ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютеризованих систем управління

Освітнього ступеня магістр

Напрям (спеціальність) 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри

Литвиненко О.Є.

«_____» _____2022 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи (проєкту)

Ященка Костянтина Анатолійовича
(П.І.Б. випускника)

1. Тема роботи (проєкту): «Геоінформаційна система аналізу і візуалізації даних трекінгу джерел поповнення енергії»

затверджена наказом ректора від «16» 09 2022р. № 1530/ст

2. Термін виконання роботи (проєкту): з 05 вересня 2022 року по 30 листопада 2022 року

3. Вихідні дані до роботи (проєкту): завдання на кваліфікаційну роботу, аналіз даних трекінгу джерел поповнення енергії, візуалізація даних трекінгу джерел поповнення енергії, концептуальна модель *ExIFO2*, мова програмування *C++*, мультимедійна бібліотека *SFML*.

4. Зміст пояснювальної записки:

1) Принципи побудови геоінформаційних систем.
2) Проєктування геоінформаційної системи аналізу та візуалізації даних трекінгу джерел поповнення енергії.
3) Розробка геоінформаційної системи.

5. Перелік обов'язкового ілюстративного матеріалу:

1) діаграма основних класів;
2) робота програмного методу *drawMap()* (схема алгоритму);
3) вікно програми для вибору авто;
4) вікно програми для показу цифрової мапи;
5) результати аналізу даних трекінгу джерел поповнення енергії.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Написати перший розділ пояснювальної записки	05.09.22 – 20.09.22	
2	Сформувати попередню структуру системи	21.09.22 – 25.09.22	
3	Написати другий розділ пояснювальної записки	26.09.22 – 06.10.22	
4	Розробити систему	07.10.22 – 20.10.22	
5	Написати третій розділ пояснювальної записки	21.10.22 – 01.11.22	
6	Завершити оформлення пояснювальної записки	02.11.22 – 12.11.22	
7	Підготувати графічні матеріали та презентацію	13.11.22 – 21.11.22	

7. Дата видачі завдання: «05» вересня 2022 р.

Керівник кваліфікаційної роботи (проєкту) _____ Халімон Н.Ф.

(підпис керівника) (П.І.Б.)

завдання прийняв до виконання _____ Ященко К.А.

(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Геоінформаційна система аналізу і візуалізації даних трекінгу джерел поповнення енергії» містить: 93 с., 27 рис., 20 літературних джерел.

ГІС, ДЖЕРЕЛА ПОПОВНЕННЯ ЕНЕРГІЇ, ТРЕКІНГ, КОНЦЕПТУАЛЬНА МОДЕЛЬ, ВІЗУАЛІЗАЦІЯ, ЕЛЕКТРОКАР, ЗАРЯДНА СТАНЦІЯ.

Об’єкт дослідження – географічна інформаційна система для збору, зберігання, аналізу та графічної візуалізації просторових (географічних) даних та пов'язаної з ними інформації про необхідні об'єкти.

Предмет дослідження – географічна інформаційна система (геоінформаційна система) аналізу і візуалізації даних трекінгу джерел поповнення енергії.

Мета кваліфікаційної роботи – створення геоінформаційної системи аналізу і візуалізації даних трекінгу джерел поповнення енергії.

Результати виконання кваліфікаційної роботи: розроблено геоінформаційну систему аналізу і візуалізації даних трекінгу джерел поповнення енергії.

Практичне значення: результати, отримані в кваліфікаційній роботі, можуть бути використані в процесі вибору джерела поповнення енергії (зарядної станції) по даним трекінгу відносно положення та характеристик авто (споживача) та візуалізації через *GUI*, що включає графічні кнопки, інформаційні панелі та цифрову мапу.

Прогнозні припущення щодо розвитку об’єкта дослідження: покращення можливостей за рахунок подальшого доопрацювання алгоритмів тестування сумісності джерел поповнення енергії, покращення візуальної структури *GUI*, застосування реальних *GPS*-сервісів, додавання більшої кількості моделей електромобілів для перевірки, підтримка кросплатформності.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

SFML – *Simple and Fast Multimedia Library*

*VS*2017 – *Visual Studio* 2017

ГІС – географічна інформаційна система

ДПЕ – джерело поповнення енергії

MPEI – макет розташування елементів інтерфейсу

ВСТУП

Люди завжди приділяли особливу увагу ремеслу навігації, так як одним з перших необхідних дій для поповнення ресурсів та й існування загалом було дослідження нових територій, спочатку локально, але потім перетікаючи в більш глобальний масштаб. Часом це здійснювалось з цікавості, часом з необхідності, але в результаті відкривання нових земель вело до розповсюдження та змішування культур та відкривало нові горизонти й можливості для взаємодії та обміну досвідом багатьох народів та цивілізацій. Люди покращували способи орієнтування та пошуку потрібного місця призначення. До способів орієнтування по «поверхневим» природним чинникам, як от положення зір на нічному небі чи сторона росту моху на стовбурі дерева, додаються нові, в тому числі й з використанням штучних інструментів, створених людиною. Ці інструменти, такі як найпростіші компаси та мапи, що в тандемі дають непогану систему навігації, пройшли довгий шлях відточування точності використання, способів створення та з часом зазнали сильної модернізації. Тепер, завдяки електроніці сучасна навігація майже повністю перейшла в цифровий вигляд, а враховуючи також Інтернет, яким охоплена велика частина Земної кулі, синхронізація та передача даних, які потрібні для орієнтування по тих чи інших критеріях, будь то звичайна мапа автомобільних доріг чи аналітично вивірена мапа споживачів певних товарів вжитку, для маркетологів здійснюється майже миттєво, а можливості такого орієнтування доступні навіть рядовому користувачеві. Важливу роль в цифровій навігації відіграє спеціальне програмне забезпечення, яке орієнтоване на відповідне оброблення та зручне подання користувачеві географічної інформації різного типу й призначення – *GIS* (*Geographic Information System*, ГІС – геоінформаційна система).

Даний дипломний проєкт виконується з метою створення власної ГІС, що орієнтована на пошук підходящого джерела поповнення енергії на основі даних трекінгу для певного споживача електроенергії, наприклад електромобілю. ГІС зможе зекономити кінцевому користувачеві багато часу, нервів, та, що головне –

лаконічно надати потрібну інформацію про найближче від користувача сумісне джерело для підзарядки електромобіля.

Об’єкт дослідження – географічна інформаційна система для збору, зберігання, аналізу та графічної візуалізації просторових (географічних) даних та пов'язаної з ними інформації про необхідні об'єкти.

Предмет дослідження – географічна інформаційна система (геоінформаційна система) аналізу і візуалізації даних трекінгу джерел поповнення енергії.

Мета кваліфікаційної роботи – створення геоінформаційної системи аналізу і візуалізації даних трекінгу джерел поповнення енергії.

ГІС-технології застосовують для комплексного вивчення природно-економічного потенціалу великих регіонів, проектування природних магістралей, забезпечення життєдіяльності людини, при використанні нетрадиційних джерел енергії. Через поступову відмову людей від нафтопродуктів спостерігається значний ріст використання засобів пересування на основі електродвигунів. Разом з поширенням електротранспорту додаються нові удосконалені способи заряджання через спеціалізовані станції зарядки, тому тема кваліфікаційної роботи «Геоінформаційна система аналізу і візуалізації даних трекінгу джерел поповнення енергії» є актуальною.

Для досягнення вказаної мети, виконано такі завдання:

– оглянути визначення та застосування ГІС;
– оглянути предметну область ГІС;
– здійснити проєктування ГІС;
– сформувати алгоритми роботи ГІС;
– обрати компоненти для розробки ГІС;
– розробити ГІС.

Для виконання наведених пунктів кваліфікаційну роботу поділено на три розділи. Перший розділ присвячено огляду предметної області та постановці задачі дослідження. У другому розділі розглянуто проєктування системи. Третій розділ присвячено розробці системи.

РОЗДІЛ 1
ПРИНЦИПИ ПОБУДОВИ ГЕОІНФОРМАЦІЙНИХ СИСТЕМ

1.1. Основні компоненти геоінформаційних систем

Геоінформаційні системи являють собою комплекси засобів, які сприяють комфортному та ефективному використанню різного типу даних для орієнтуванню по цифрових картах [1]. Такі системи можуть включати різні можливості для поліпшення досвіду використання, тобто можуть бути можливості збору, зберігання, аналізу та візуалізації геоданих.

Збір геоданих зазвичай використовується для постановки початкових значень для різноманітних алгоритмів обробки ГІС залежно від її направленості. Крім того, збір часто здійснюється не просто з інших ланок системи (якщо такі наявні), а й з інших джерел взагалі – зазвичай це відкриті картографічні сервіси, наприклад *Google Maps*, який має у своєму розпорядженні надвеликі масиви даних, такі як цифрові карти (схематичні), супутникові знімки та інші геодані, які додає сама компанія та її користувачі по всьому світу. Але трапляються й інші випадки, коли потрібен особливий підхід до певної території, наприклад особливу деталізацію, інші інформаційні шари місцевості (наприклад ландшафтні дані, дані флори/фауни або інші спеціальні можливості). В такому разі, особливо в разі комерційної цінності можуть застосовуватися дані із закритих джерел геоданих (які зазвичай є платними), або ж навіть доводиться замовляти та/або організовувати окремі дослідження для витягу потрібних, а головне – актуальних геоданих.

Зберігання геоданих є невід'ємною частиною подібних систем. При чому не так важливо в даному випадку – чи здійснюється таке зберігання на стороні сервера чи на стороні клієнта. Наскільки часто дані будуть оновлюватися залежить від направленості системи. Суть в тому, що необхідний мінімум даних – це зображення місцевості у зручний для користувача спосіб. Різні позначення на карті, такі як певні географічні точки, маршрути, засоби навігації (віртуальний компас і подібне)

можуть бути представлені для зручності й наочності наданих користувачеві даних. Якщо ставка робиться на глобальність системи, то й сховище даних повинно буде готове до прийняття великого об’єму даних, тому часто використовується клієнт-серверна архітектура ГІС, в основному використовуючи Інтернет для оновлення інформації та зв’язку з клієнтським додатком. Особливо це розповсюджено з відкритими системами загального використання, так як клієнтський додаток зазвичай не має достатньо пам’яті для зберігання різного типу спеціальних БД, таблиць, впорядкованих текстових документів й тд. Тому такі об’єми даних зберігаються на окремих серверах. Є потреба розробки й офлайн-версій ГІС, тобто таких, якими б можна було б користуватися й поза доступом в Інтернет. Переваги, як і недоліки таких систем очевидні: хоча користувачеві й не треба постійний доступ в Інтернет та витрачати час на оновлення, при тому дані будуть не найактуальнішими та доведеться все ж зберігати на своєму девайсі певний об’єм інформації, такі як цифрові карти та положення на них географічних об’єктів (будь то природні чи об’єкти людської інфраструктури). Також є гібридний варіант побудови ГІС, які оновлюються по Інтернету, але зберігають потрібні користувачеві дані на пристрої для можливості використання ГІС в офлайні.

Аналіз геоданих надає нові дані на основі початкових, які зберігалися хронологічно першими. Таким чином можна навіть заощадити на об’ємі пам’яті, так як деякі результати можна обраховувати лише за потреби, наприклад знаючи відстань до точки та швидкість пересування користувача можна вирахувати час прибуття (у всякому разі близький до середніх показників) до пункту призначення. Звичайно для швидкості обрахунків чи набору статистичних даних час до прибуття можна кешувати чи зберегти в окрему колонку таблиці, але це ж і займе пам’ять. Крім того, аналіз забирає певний час і зменшує обчислювальну потужність пристрою-виконавця, але якщо такі витрати оправдані та перерозподілені на правильні ланки системи (дата-центри чи часткові обчислення на інших пристроях) тоді відносно не впливає на швидкодію. Аналіз геоданих дозволяє зробити систему набагато більш інтерактивною, бо ж обчислення виконуються «на льоту», так як це робиться в цифровому компасі (який працює або на основі тих же геоданих і

обчислюється на стороні, або якщо відповідні датчики вже вбудовані у одну з ланок системи, то такі дані треба лише прив’язати до поточних розрахунків і вивести користувачеві для сприйняття). Крім того, аналіз даних в системі робить її куди універсальнішою, тому переважна більшість ГІС мають не лише суто статичний, а й динамічний інформаційний характер.

Графічна візуалізація геоданих є чи не найважливішою стороною створення ГІС щодо взаємодії з користувачем-людиною. Звісно, для кращого сприйняття, а як наслідок, і продуктивнішої роботи з системою людина повинна приймати та обробляти інформацію, що зручна для людського мислення, у вигляді абстракцій. Люди давно користуються мапами, але от сучасні технології зробили кращими роботу з ними не лише в плані аналізу даних, а й в плані візуалізації – графічного відображення. Крім основних функцій відображення самої мапи та деяких дій, зв’язаних з нюансами передавання мап з дисплею (масштабування, візуальне переміщення по карті) є й інша сторона, та, яка впроваджує в цифрові мапи додаткові, не доступні для традиційних матеріальних мап дії – від відстеження змін на мапі (збереження декількох станів мапи для подальшого порівняння) до додавання третього виміру місцевості (ландшафтні дані, відображені у вигляді *3D*-моделі). Простір для покращення базових можливостей відображення мапи з відмітками – величезний. Але так як цифрові мапи в ГІС часто надаються користувачеві саме не в «сирому» вигляді лише графічного файлу, а «в обгортці» графічного інтерфейсу, який і забезпечує людині інтуїтивно зрозумілу взаємодію з ГІС, потрібно враховувати не лише візуалізацію основних функцій, а й комфорт роботи з інтерфейсом усієї ГІС, доступної для користувача, враховуючи різні пункти налаштування та програмні меню в цілому.

Тепер, знаючи можливості ГІС, можна переходити до огляду її компонентів. Тут важливо зазначити, що сам по собі термін «геоінформаційна система» має два значення.

Геоінформаційна система в широкому значенні – це комплекс засобів для орієнтування по цифрових картах, який включає в себе широкий круг матеріальних та нематеріальних компонентів [2], а саме:

– апаратні засоби;

– програмне забезпечення;

– дані;

– виконавці;

– методи.

До апаратних засобів відноситься обладнання, на якому запущено геоінформаційну систему. Зазвичай це комп'ютерна платформа на базі ядра *Linux* або ОС *Windows*. Сам тип платформи може бути найрізноманітніший – наприклад, спеціальний централізований сервер або навіть один, або декілька (зв'язаних в мережу) звичайних настільних ПК. Вибір апаратних засобів залежить від направленості та можливих потреб при експлуатації окремої ГІС.

ПЗ для ГІС включає в себе різноманітні програмні інструменти та функції, які необхідні для функціонування ГІС (аналіз, зберігання, візуалізація тощо). Основними компонентами програмних продуктів можуть бути: інструменти для введення та оперування географічною інформацією; засоби управління даними, якими оперує ГІС – СУБД, або подібні за значенням програмні функції; інструменти підтримки просторових запитів, аналізу та візуалізації; графічний інтерфейс (GUI) для взаємодією користувача з програмними інструментами СУБД.

Дані є одним з найважливіших компонентів ГІС. Географічні дані про просторове положення, які також можуть бути оформлені в вигляді таблиць чи іншим впорядкованим чином купуються у постачальників або збираються й оформлюються самим користувачем. Під час управління просторовими даними ГІС може використовувати дані інших типів та з інших джерел.

ГІС не могла б повноцінно розроблятися та використовуватися без людського фактору – будь то розробник чи кінцевий рядовий користувач. Саме люди, які беруть участь у життєвому циклі геоінформаційних систем є виконавцями. Таким чином виконавці розробляють сам програмний продукт, відповідають за апаратне оснащення, створюють план розвитку, обирають сфери використання та спосіб підтримки ГІС, а також вводять в експлуатацію, оновлюють, користуються та в цілому взаємодіють з системою.

Методи як компонент геоінформаційної системи в широкому розумінні стосуються не окремих компонентів, а системи в цілому, а точніше конкретизації та формулюванні призначення, експлуатації та розвитку ГІС. В основному це прийняття стратегічних рішень існування системи – від проєктування до економічних рішень.

Поняття геоінформаційної системи також використовується у більш вузькому значенні – як програмного комплексу [3] (набору інструментів, продукту), що може дозволяти користувачам переглядати та/або взаємодіяти з цифровою картою місцевості, отримувати додаткову інформацію про географічні об'єкти. Тобто йдеться саме про ПЗ геоінформаційної системи. Можливі компоненти такого ПЗ було вже описано вище, але розглядаючи цей аспект детальніше, можна додати декілька пунктів до загального переліку складових ПЗ ГІС. Програмні компоненти ГІС можуть включати:

– програмне забезпечення для малювання карт;

– пакети геометричного моделювання;

– пакети візуалізації даних;

– пакети 3D-моделювання та симуляцій;

– системи керування базами даних ГІС;

– пакети для аналізу даних ГІС;

– статистичні пакети.

Також, залежно від направленості та потреб для виконання поставлених задач, ГІС класифікують за різними критеріями [4]:

За територіальним охопленням:

– глобальні ГІС;

– субконтинентальні ГІС;

– національні ГІС;

– регіональні ДВС;

– субрегіональні ГІС;

– локальні чи місцеві ГІС.

За функціональністю:

– повнофункціональні;

– ГІС для перегляду даних;

– ГІС для введення та обробки даних;

– спеціалізовані ГІС.

За проблемно-тематичною орієнтацією:

– земельно-кадастрові;

– екологічні і природокористувальницькі;

– інженерних комунікацій і міського господарства;

– надзвичайних ситуацій;

– навігаційні;

– соціально-економічні;

– геологічні;

– транспортні;

– торгово-маркетингові;

– археологічні;

– військові;

– інші.

При тому до цього критерію даної класифікації перерахування на пункті «інші» точно не закінчуються, так як сфери застосування ГІС розширюються та комбінуються. Це ще один доказ того, що ГІС є актуальними зараз, а ріст кількості їх направленостей та загальний розвиток технологій (в тому числі різних способів збору й обробки геоданих) сприяє лише подальшій актуальності подібних систем.

1.2. Аналіз ринку геоінформаційних систем

AutoCAD Map 3D є перехідною ланкою між ГІС та САПР (системою автоматизованого проєктування – для автоматизації технологічного процесу проєктування виробу, результатом якого є комплект проєктно-конструкторської документації, достатньої для виготовлення та подальшої експлуатації об'єкта проєктування). Не варто забувати, що компанія *Autodesk* має великий досвід

розробки допоміжних систем проєктування та розробки, а ГІС, в свою чергу, може витупати як важлива складова подібних процесів. Як вже зрозуміло, головною особливістю *AutoCAD Map 3D* є найтивна інтеграція з сервісами *Autodesk*. Таким чином, можливості ГІС поєднуються з великим інструментарієм редагування. Наприклад, якщо потрібно скласти план розташування житлового комплексу, для цього треба гарно розуміти параметри території, на якій відбуватиметься будівництво. Маючи картографічні знімки цільової території, *AutoCAD Map 3D* дозволить в самій програмі користуватися САПР-направленними інструментами: крапки, лінії, полігони – все для зручного проєктування користуючись вбудованими інструментами, без потреби «жонглювати» даними між різними застосунками. Також вказується про наявність інструментів просторової прив'язки та інших корисних функцій.

Крім того, не можна забувати про вельми корисні інструменти аналізу даних та менеджменту. Це може стати в пригоді при різноманітних розрахунках у проєктуванні, основаному в тому числі на геоданих: наприклад, на певній ділянці планується кинути трубопровід. Із зрозумілих причин, небажано щоб над трубопроводом велися інші будівельні роботи, тому при проєктуванні користувачу не варто турбуватися про поетапне зазначення відповідної «неробочої» зони навколо трубопроводу – через спеціальний інструментальний буфер при проєктуванні мережі труб на план буде автоматично наноситися відповідна зона в певному радіусі навколо неї. Зрозуміло, що в подібних випадках маємо справу з геопросторовим аналізом, так як можуть бути певні нюанси при проведенні різних робіт на території. Також, можливо створювати та вносити окремі набори даних проєкту, наприклад прокладаючи ті ж труби, варто лише раз налаштувати їх параметри – сегментованість, діаметр, товщину стінок, матеріал, інші особливості, і при роботі з планом у *AutoCAD Map 3D.* Так, наприклад, можна буле відразу порівняти матеріали труб та грунту, і при необхідності скорегувати процес будівництва. І це лише один приклад переваг такої гібридності ГІС системи й САПР.

В цілому, *AutoCAD Map 3D* безперечно вартий уваги, особливо при використанні в інженерії, пов’язаній з територіальним збором даних. Крім комерційної пропонується й безкоштовна пробна версія *AutoCAD Map 3D* для студентів й викладачів.

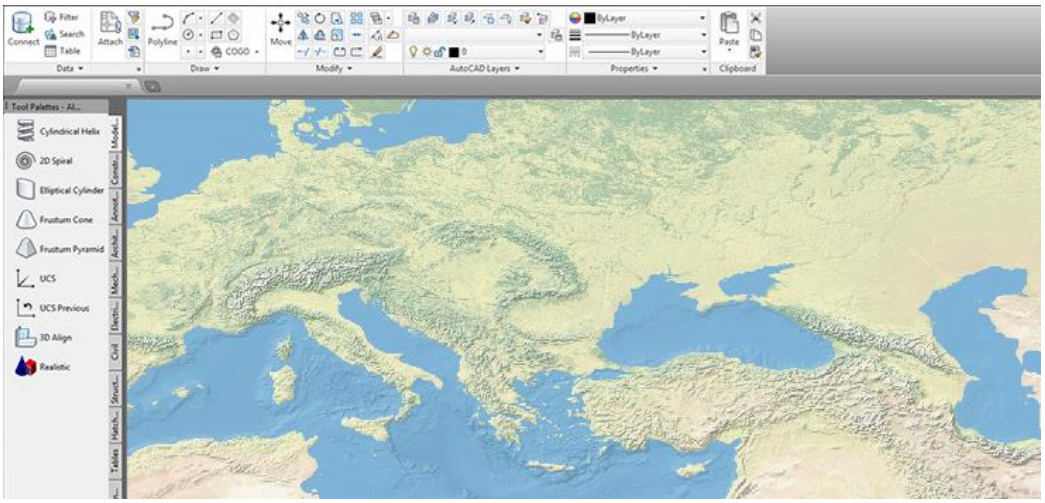

Рис. 1.1. Інтерфейс *AutoCAD Map 3D*

Переваги:

– інтерфейс та сервіси будуть знайомими для користувачів продуктів *Autodesk*;

– стрічковий інтерфейс з інтуїтивно зрозумілим розташуванням та логічно оформленою організацією меню;

– вдале поєднання ГІС та САПР;

– наявні інструменти для роботи з хмарами точок *Surface* та *LiDAR*;

– хороший загальний набір інструментів.

Недоліки:

– зависока вартість ліцензії;

– підтримка продукту не на висоті;

– високий поріг входження для тих, хто раніше не працював з продуктами *Autodesk*;

– спеціалізовані інструменти вельми обмежені при користуванні.

MapInfo Professional виконує основне завдання подібних систем – зв’язує географічні та візуальні дані, і виконує це завдання на всю свою ціну ліцензії, яка

є немалою. Девіз даного програмного продукту – «*location intelligence*», що можна перевести як «інтелектуальне визначення місцезнаходження». Детальне картографування, аналіз даних, виявлення закономірностей та тенденцій – ці аспекти й роблять успішними *MapInfo Professional* на ринку. *MapInfo GIS Suite,* який входить до *MapInfo Professional* дозволяє створювати, отримувати доступ та керувати геопросторовими активами, візуалізувати бізнес-аналітику та дані про клієнтів, а також обмінюватися високоякісними інтерактивними мапами – швидко та легко.

Візуалізація це ключ до продуктивності. І це видно, якщо оглянути графічний інтерфейс застосунку. При масштабуванні мапи іконки не зливаються, текст залишається максимально чітким. Пункти меню розташовані помірно один відносно одного, використовуються багато графічних позначень в пунктах меню, інтерфейс по першому досвіду схожий на відповідні меню стандартного інструмента ОС *Windows – Paint*. Таким чином знову зустрічається стрічковий вид розташування. Таким чином, наприклад керування таблицями та запити доступні на окремих вкладках. В дописі до оновлення вказано, що в нових версіях покращене картографічне виведення, проведена робота над зручністю виведення позначок, легенд (коротких інформативних описів) та масштабовими лінійками. Цікавою функцією видається можливість паралельного співставлення інформації з мапи.

Крім гарної та інформативної візуалізації сильною стороною програми є також аналіз геоданих. *MapInfo Professional* особливу увагу надає розташуванню, тобто на основі даних про географічне положення, які співставляються з різними джерелами. До прикладу, зі спеціальними ГІС-картами страхових компаній *MapInfo Professional* може надати рекомендації щодо страхових ризиків при тому чи іншому розташуванні обраного об’єкту. Або на основі статистичних даних щодо списку предметів частого вжитку у людей в певній житловій зоні. Так програма допомагає виявляти різні географічні закономірності, які при грамотному підході можуть сильно підсобити при організації справ, особливо це стосується бізнесу.

Важливою функцією є також геокодування, так як в такому разі важливо знати місцезнаходження потенційних клієнтів та конкурентних об’єктів.

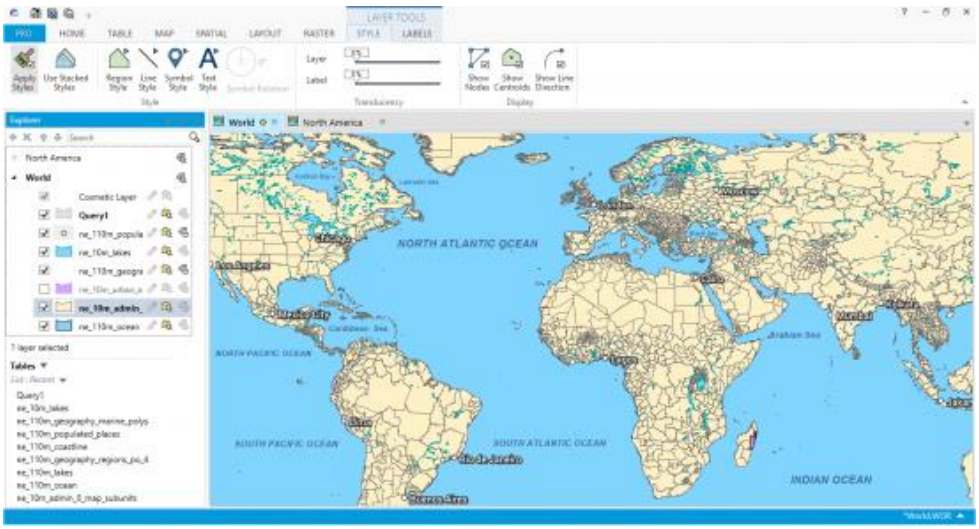

Рис. 1.2. Інтерфейс *MapInfo Professional*

Також дана ГІС підтримує додаткові інструменти, такі як *LiDAR*, дистанційне зондування і тд. Із можливих незручностей можна зазначити те, що відносно важко добавляти власні дані в *MapInfo Professional*, програма також підтримує спеціальне своє розширення файлів геоданих «*.TAB*», але цей формат не є універсальним. Навіть зі сумісністю *FME* складно додати *GeoTIFF* або необроблені супутникові зображення. Все ж не можна сказати, що в даній ГІС бракує інструментів, але все ж більшість наявні й в інших системах. Сильна сторона цієї – бізнес-аналітика на основі геоданих, та й видно, що інструментарій на це й розрахований. Позитивною стороною даного застосунку є не лише його описані можливості, а й інформативне та обширна спільнота. *Li360* — це ГІС-спільнота *MapInfo.* Там наявні широкі роз’яснення по документації, контенту користувача і відповіді на часті питання по продукту.

Переваги:

– простота використання інтерактивних мап;

– потужні служби адресації та геокодування;

– паралельне зіставлення для покращення ефективності візуалізації;

– обширна спільнота програми.

Недоліки:

– направленість лише на бізнес-аналітику;

– проблеми при додаванні власних даних в програму;

– підтримка інших форматів файлів з геоданими залишає бажати кращого;

– відсутність підтримки хмарних технологій в програмі такої спрямованості;

– висока ціна.

ArcGIS Desktop − найвідоміша платна ГІС. Перевагою даної системи є чудова масштабованість. Це означає, що *ArcGIS Desktop* можна налаштувати під себе використовуючи різні налаштування та, найголовніше – велику кількість програмних розширень для даної ГІС, які можуть ще й комбінуватися між собою для досягнення необхідних цілей для конкретного клієнта. Надається велика кількість детально описаної документації та екстраширокий вибір фірмових інструментів *ArcToolbox*. Підтримується *3D*-відображення карт та онлайн сервіси для редагування й перегляду за потреби. Крім того *ArcGIS Desktop* має ще один «козир в рукаві» – найтивним джерелом геоданих різного типу та об’єму є масивне онлайн-сховище даних *ArcGIS Online*. Це добре організоване відкрите джерело, що є основою геоданих для багатьох інших ГІС. Також слід зазначити порівняно комфортне середовище для редагування, але яке не має якихось особливостей щодо інших подібних інтерфейсів.

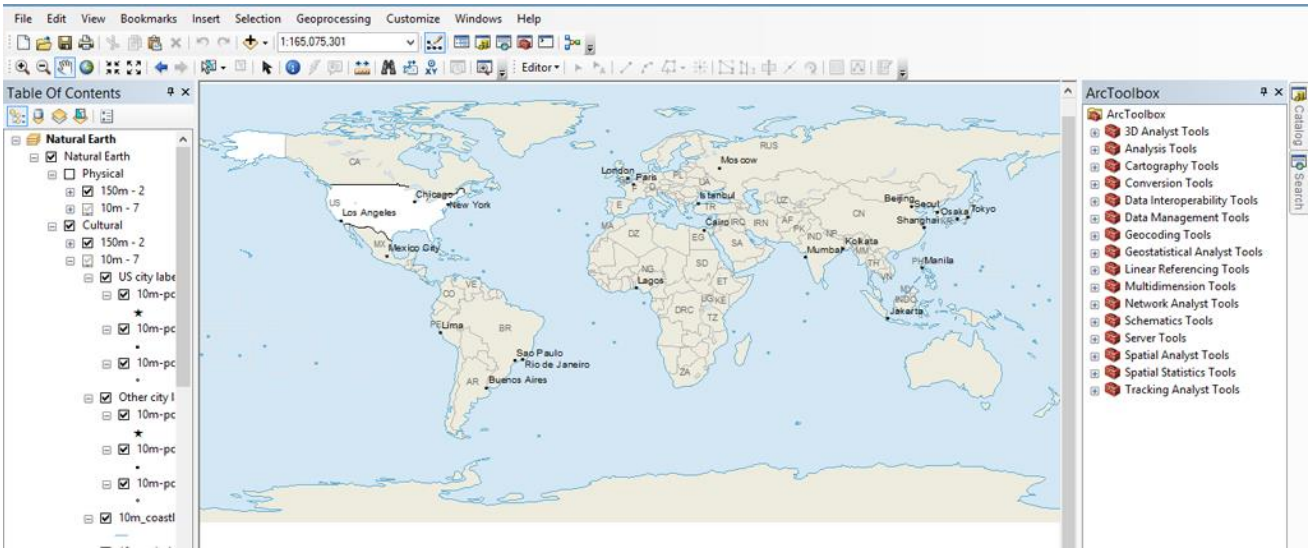

Рис. 1.3. Інтерфейс *ArcGIS Desktop*

При тому дана система має й недоліки. Наприклад, сумісність з певними типами даних залишає бажати кращого, і хоча з часом оновлення поправили загальну сумісність, все ж такій популярній ГІС не завадило б впровадити більшу підтримку різних форматів геоданих. Також до суттєвих мінусів можна віднести високу ціну повної ліцензії. Хоч *ArcGIS Desktop* і має достатній набір функцій, але все ж для повноцінного використання потенціалу продуктів *ArcGIS* потрібно мати доступ до всього інструментарію, що дозволяє лише окремий тип ліцензії – *ArcInfo.* При базовій ліцензії багато фірмових інструментів просто недоступні. Крім того, багато дійсно корисних розширень для даної ГІС не є безкоштовними, що знову ж підвищує суму витрат на володіння системою.

Переваги:

– висока розповсюдженість системи у закладах освіти з відповідним напрямом навчання;

– стандартний інтерфейс зі стандартним комфортом;

– висока масштабованість можливостей при розширеннях;

– наявність повного набору інструментів топології та редагування;

– наявність *ArcGIS Online* для підтримки мобільних застосунків та веб-карт.

Недоліки:

– висока вартість використання ГІС;

– базовий рівень ліцензії надає обмежений набір інструментів;

– недостатня сумісність та підтримці нестандартних форматів.

QGIS 3 відома серед професіоналів та аматорів. *QGIS 3* – це безкоштовна геоінформаційна система з відкритим початковим кодом. Як можна здогадатися, це вже третя велика ревізія даної системи, а враховуючи її безкоштовність, то стає зрозуміло, що по функціям вона не уступає своїм платним аналогам, так як публіка продовжує користуватися та підтримувати *QGIS 3*. Спільнота цієї ГІС дуже велике, і має як ніколи тісні зв’язки та вплив на розробників.

QGIS 3 підтримує *3D*-візуалізацію мап за замовчуванням, тобто без додаткових плагінів. Враховуючи, що минула ревізія підтримувала *3D* опосередковано і лише з відповідними плагінами, то це видається як великий крок

вперед при розвитку системи. Також *QGIS 3* працює над тією областю подібних систем, яка залишається без належної уваги у більшості конкурентів – в саме над інструментами редагування стилів візуалізацій. Так, звісно розташування меню, наявність потрібних інструментів саме для обробки геоданих та стабільна робота системи дійсно важлива, але не треба забувати і про деталі візуалізації, так як переважна більшість ГІС використовується в тому числі для кінцевого а проміжного аналізу людським зором, а значить геодані треба не просто зручно та правильно відредагувати чи розрахувати – а також й подати так, щоб кінцевий користувач міг сприйняти надану від ГІС інформацію. *QGIS 3* надає окремі спеціальні інструменти для налаштування стилей та кольорових схем представлення мап та геоданих в цілому. По суті, це міні-графічний редактор для схематичних позначень та інших даних, зв'язані з мапами. Наприклад, є інструмент «альфа-слайдер» для зміни прозорості того чи іншого об'єкта на мапі, або інструмент під назвою «віджет кольорової рампи», який дозволяє керувати кольоровими схемами відображення цілих проєктів, редагувати, та зберігати такі схеми в шаблони для подальшого користування.

Розробники також постаралися над швидкодією системи, особливо функцій, що використовуються найчастіше: панорамування, зум, завантаження інструментів та показ зображення мапи з відмітками. А розумне кешування дозволяє пришвидшити рендеринг та перемальовку.

Також дуже плідний підхід був здійснений до функцій розміщення міток. Якщо традиційний підхід полягає в створенні анотацій для розміщення та зміни налаштувань шару з мітками, що часто призводить до накопичення шарів, зниження швидкодії та вигляду мапи важкого для візуального сприймання, то в даному випадку *QGIS 3* надає можливість використання одного універсального шару, який реагує на мітки як на фігури на шаховій дошці. Так мітки можна переміщати вручну, стилізувати, приховувати окремі або групи міток. Таким мінімалістичним підходом дана ГІС відмовляється від анотацій на користь більш інтуїтивно зрозумілого механізму взаємодії з мітками на мапі при редагуванні.

QGIS 3 додає нестандартні для своєї області використання інструменти, наприклад, одна з найкорисніших – це інструмент відслідковування правок від декількох користувачів. Це дуже цінна можливість, яка дозволяє ефективно користуватися ГІС групою співробітників. Також щодо нестандартних інструментів можна відмітити часткове використання елементів САПР при редагуванні, що має великі можливості при розвитку такого направлення в інструментарії.

Слід зазначити й перехід в новій ревізії на інших стандартний формат файлу для виведення з ГІС – стався перехід зі специфічного шейпфайлу на геопакет (*geopackage*) формату «.*GPKG*». Це важливо тим аспектом, що «.*GPKG*» це формат автономної безсерверної бази даних *SpatiaLite*, і цей тип файлу може зберігати все що завгодно – від векторної графіки до навіть цілих розширень. Це означає набагато більшу універсальність користування даними, що надає *QGIS 3*.

Ще одне нововведення – це можливість фонового опрацювання при задіянні певних інструментів. Наприклад, розрахунок шару з координатними точками може здійснюватися в фоні, а користувач тим часом зможе далі працювати в ГІС, не чикаючи кінця обробки.

Щодо сумісності, то ще одна перевага – це підтримка проєктів, зроблених в минулій ревізії даної ГІС. Це означає, що користувачі *QGIS 2.18* зможуть імпортувати свої роботи до *QGIS 3* без додаткових труднощів.

Дана ГІС також робить ставку на величезну кількість додаткових плагінів, що підтримується та створюються в більшій мірі сторонніми ентузіастами-розробниками для розширення можливостей *QGIS 3*. Хоча варто зазначити, що плагіни для попередніх ревізій цієї ГІС початково не сумісні з *QGIS 3*. Але знову ж видно велику любов та підтримку спільноти програми : кожного дня ними старі плагіни переписуються для нової версії. Але і офіційні розробники заявляють, що в новій ревізії ГІС більшу половину програмного коду інструментарію було переписано на *C++* майже з нуля.

Також не можна не згадати про таке нововведення як можливість графічного моделювання без плагінів. Тут особливість в тому, що тепер не аналіз працює на

візуалізацію, як в цифрових мапах, а візуалізація працює на аналіз. Це схоже на програмування через графічні блоки через *Blueprints* в мультимедійному рушієві *Unreal Engine 4*. Тільки в даному разі вигляд графічних блоків з даними допомагає користувачеві спланувати певний процес моделювання, де будуть початкові блоки введення даних, проміжні для обробки, роботу яких можна налаштовувати в редакторі та блоки виведення. Це робота не з кодом, скриптами чи навіть таблицями, а куди зручніша з візуальним моделюванням обробки даних.

Оглянувши основні аспекти даної ГІС можна виразити припущення, що якщо *ArcGIS* це найкраща платна ГІС на ринку, то *QGIS 3* це безперечно прямий безкоштовний конкурент зазначеної під ліцензією типу *GNU General Public License* з відкритим початковим кодом.

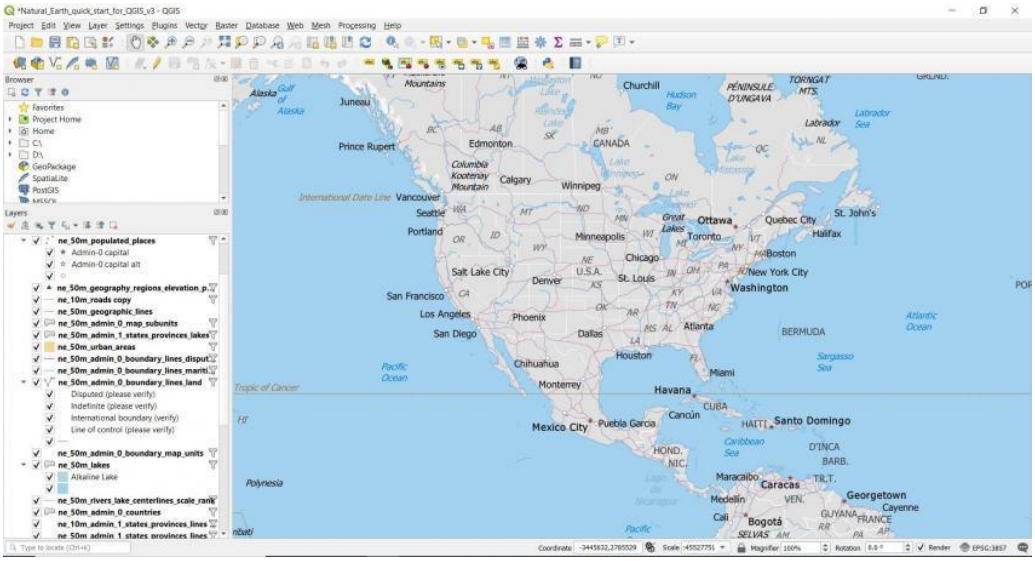

Рис. 1.4. Загальний вигляд інтерфейсу *QGIS 3*

Переваги:

– безкоштовність ліцензії та підтримки продукту;

– потужна вмонтована можливість роботи з *3D*-відображенням;

– інноваційні в своїй області інструменти аналізу, візуалізації та редагування;

– величезна відкрита база геоданих від користувачів;

– швидка та якісна підтримка від розробників та спільноти;

– відкритий програмний код;

– безкоштовність та відкритість додаткових плагінів;

Недоліки:

– відсутність деяких вузьконаправлених спеціалізованих інструментів аналізу;

– проблеми зі стабільністю *3D* візуалізацій та *LiDAR*;

– обмежена високорівнева веб-картографія та підтримка веб-застосунків;

– відстає від конкурентів у нових технологіях (ШІ, *Big Data*, машинне навчання тощо).

1.3. Постановка задач дипломного дослідження

Для створення ГІС було обрано дві задачі: аналіз даних трекінгу джерел поповнення енергії та візуалізацію результатів та інтерактивних елементів через простий *GUI*. В поєднанні ці задачі мають складати інтуїтивно зрозумілу геоінформаційну систему, що надає користувачеві обране за алгоритмами ГІС джерело поповнення енергії на цифровій мапі та візуалізує маршрут за наданими зовнішніми даними *GPS*-сервісів між цим джерелом та точкою місцезнаходження споживача. Тобто завданням розроблюваної ГІС є підбір найкращого по характеристикам джерела поповнення енергії на мапі відносно споживача й зручне мінімалістичне графічне виведення цієї інформації користувачеві.

Наприклад, для пошуку певних географічних точок користувачеві потрібно або достатньо гарно орієнтуватися на певній місцевості, або ж застосовувати допоміжні засоби для орієнтування – від спілкування з місцевими до використання спеціальних навігаційних інструментів. В еру цифрових технологій та бездротових мереж користувач часто має з собою різні портативні комунікатори, а враховуючи розповсюдженість та доступність сучасних пристроїв, це зазвичай смартфон чи ноутбук. Але також в життя сучасної людини входять не лише подібні апарати, а й більш складні та масивні, які виконують зовсім інші практичні задачі – одними з таких електрокари, принцип роботи яких, звісно, багато в чому відрізняється від звичайної портативної електроніки, але має дещо спільне – навіть таку складну

електронно-механічну систему як електромобіль потрібно теж заряджати електроенергією, а для більшості сучасних моделей таких авто живлення від побутової мережі є вже не просто небезпечним та невиправдано довготривалим, а навіть непередбаченим виробником. Отже, виникає логічна думка: якщо електрокари мають обмежену кількість заряду, то треба завжди знати, де можна поповнити заряд елемента живлення безпечно та доступно. Дана ГІС буде чудовим програмним компаньйоном для допомоги у виборі такого джерела поповнення енергії у вигляді спеціальної зарядної станції.

Фактично, під час аналізу здійснюється порівняння двох видів характеристик джерел поповнення енергії: технічних та просторових, які надаються в тому числі попереднім трекінгом джерел. При тому порівняння здійснюється не лише між самими джерелами, а й щодо електрокара користувача теж, назва якого вказується з переліку користувачем в ГІС, тобто здійснюється перевірка на сумісність з джерелом, яка теж входить до задачі аналізу.

Під час візуалізації здійснюється безпосередня відмальовка цифрової мапи, елементів керування ГІС та результатів аналізу джерел поповнення енергії для конкретного електромобіля. В підсумку на мапі місцевості позначається підходяще джерело поповнення енергії відносно споживача та на основі просторових даних здійснюється візуалізація маршруту від точки-споживача до точки-джерела.

1.4. Висновки до розділу

У першому розділі дипломного проєкту були описані основні компоненти геоінформаційних систем. На основі такого опису можна зробити висновок, що для поточної ГІС слід застосувати такі процеси:

– зберігання;

– аналіз;

– візуалізація.

Тобто даний проєкт має зосереджуватися на локалізованій обробці та подачі даних, без онлайн-залежності, тим самим обмежуючи та спрощуючи розуміння основних процесів, які будуть проходити під час життєвого циклу ГІС.

Крім того був проаналізований програмний ринок ГІС. До уваги бралися різні аспекти схожих систем, особлива увага уділялася пошуку переваг та недоліків таких продуктів. Важливим також стала ремарка про інтуїтивно зрозумілий інтерфейс користувача в більшості програмних ГІС та можливість завантаження цифрових мап. Взято до уваги й мінімалізм панелей та меню керування, який подекуди зустрічався в *GUI* програм.

Крім того було зроблено висновки щодо розробки геоінформаційної системи:

– візуалізацію треба проводити у 2D-просторі задля простоти сприйняття користувачем та меншого витрачання апаратних ресурсів;

– так як в частини переглянутих аналогів графічний інтерфейс відчувається «перевантаженим», варто використати мінімалістичний підхід при створенні власного GUI;

– для використання ГІС не обов’язково перейматися про створення функції збору геоданих, так як їх можна створити/зібрати окремо самому, або зі сторонніх джерел;

– корисній системі необов’язково бути з громістким інструментарієм – варто зосередитися на основних функціях ГІС;

– для максимально комфортної взаємодії з ГІС користувачеві потрібна цифрова мапа;

– варто більше орієнтуватися на офлайн-режим роботи ГІС, але в поєднанні з даними GPS та іншими зовнішніми джерелами інформації.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ГЕОІНФОРМАЦІЙНОЇ СИСТЕМИ АНАЛІЗУ ТА ВІЗУАЛІЗАЦІЇ ДАНИХ ТРЕКІНГУ ДЖЕРЕЛ ПОПОВНЕННЯ ЕНЕРГІЇ

2.1. Проєктування основних компонентів геоінформаційної системи

2.1.1. Основи проєктування геоінформаційної системи

Для проєктування основних компонентів ГІС потрібно підійти до цієї задачі з боку концепцій, іншими словами створити концептуальну модель. Концептуальна модель – це тип абстракції, що використовує логічні концепції та приховує деталі фактичної внутрішньої структури та нюансів роботи системи. Концептуальні моделі стають все більше громісткими, відповідно до вимог, які має виконувати спроєктована по ним система. В певних напрямках розробки навіть порівняно надійні і популярні концептуальні моделі можуть давати не наскільки потрібні результати, особливо це стосується роботи з об’ємами даних, обробка яких націлена саме на якнайточніше відтворення певних природних явищ чи характеристик, або ж роботи з даними, де точність відтворення є основним, або одним з основних факторів. Одна справа працювати в текстовому редакторі чи здійснювати обмежене обчислення через додавання, віднімання чи множення цілих невеликих чисел, але навіть при діленні в таких, здавалось би, легких умовах, система, на якій здійснюється обчислення стикається з проблемою представлення точності. І в разі ітеративно об’ємних (через накопичувальну похибку) розрахунків (не говорячи вже про переповнення) розумно піти на компроміси, такі як округлення, або навіть додаткові способи компенсації похибки. Отже постає питання про те, чи варто взагалі витрачати ресурси на точність результату, якщо в переважній більшості випадків існує певний діапазон потрібного значення, в якому відхилення вважаються несуттєвими, а бо навіть взагалі ігноруються. Варто відповісти, що завжди потрібно прагнути до ідеального результату та максимально

цілеспрямовано покращувати способи розв’язання завдань. З практичного боку, думаю, що завжди треба зважати на фактичні чинники, такі як обмеженість таких ресурсів, як технічних, наприклад пропускна здатність певних компонентів системи чи ліміти в роботі елементів живлення, від яких система й працює, а значить треба йти на компроміси, щоб досягати може не завжди цілком ідеального, але результату, явно відповідаючому вимогам, які були поставлені до системи ще на етапі раннього проєктування. Але, при всьому тому прагненні до ідеальних результатів врахування неточностей може бути корисним.

2.1.2. Концептуальна модель стандарту ExIFO2

За своєю суттю, *ExIFO2* це розширене поєднання семантичних моделей *IFO2* та *ExIFO*, обидві з яких є розширеннями моделі *IFO*.

IFO (*Innovation-Foresight-Other processes model*, модель інноваційно-передбачуваних процесів) [5] – концептуальна модель, що являє собою формально визначену модель бази даних, яка поєднує фундаментальні принципи семантики бази даних узгодженим способом. Модель надає механізми представлення структурованих об’єктів та функціональних зв’язків між ними. Пристосована для логічного представлення даних в БД (базах даних).

IFO2 – доповнена наступна версія концептуальної моделі *IFO*. В основному додано два доповнення. Перше – сформовано та формалізовано чітке визначення ідентифікатора об’єкта, при тому ідентифікатор не залежить від значення об’єкта (для цього керовані елементи перевизначаються для врахування парадигми об’єкта). Друге – введення понять альтернативи, композиції та групування для побудови складних об’єктів.

ExIFO – модель, яка розширює концептуальну модель *IFO* тим, що вводить високорівневі примітиви щоб промоделювати нечіткі сутності [6]. Крім того, дана модель дозволяє визначати нечітку інтерпретацію логічних операторів *AND, OR, XOR*.

Щодо *ExIFO2*, то це модель даних, представлення якої здійснюється на основі графів. Можна сказати, що *ExIFO2* ще й орієнтований граф, що має різні види ребер та вершин, які ілюструють різні об’єкти (атомарні, складні) та зв’язки. В цій моделі враховуються два типи невизначеності: на рівні атрибутів та на рівні об’єкта та класу.

Таким чином, невизначеність на рівні атрибутів означає, що деякі атрибути можуть мати невизначені значення. При тому, деякі об’єкти можуть мати екземпляри з неповною приналежністю до типу цього об’єкту, тобто оцінювати власну приналежність за значенням від 0 до 1. Також, це застосовно й до відношення клас-суперклас (або клас-підклас). Від *IFO2* тут явно присутні поняття явного визначення ідентифікатора та перевизначення для цього елементів моделі, а від *ExIFO* – концепції альтернативи, композиції та групування складних об’єктів.

В графічному представлені графа концептуальна модель *ExIFO2* має наступні основні представлення.

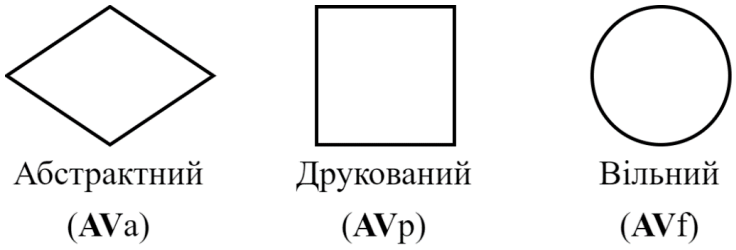

Рис. 2.1. Атомарні типи

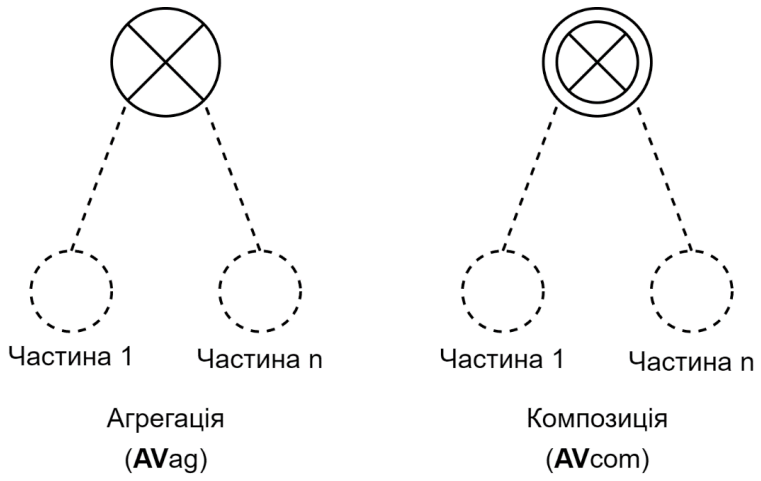

Рис. 2.2. Типи агрегації та композиції

Рис. 2.3. Типи групування, колекції та альтернативи

До атомарних (неподільних, елементарних) об’єктів належать абстрактні об’єкти, об’єкти для друку та так звані «вільні» об’єкти. Щодо неатомарних, тобто складних типів, то вони формуються за допомогою набору конструкторів.

Ці дві групи типів (атомарний та складний) є основними в моделі *ExIFO2*. Тепер, можна розглянути ці типи детальніше.

З атомарних типів, абстрактні зумовлені тим, що мають відповідати тим об’єктам, які в реальному житті не мають власної базової структури. Тип для друку відповідає вже визначеним типам, які мають на меті підтримувати введення та виведення інформації, іншими словами це тип, який відповідає за простий обмін даними з користувачем. Вільний тип відповідає за ті сутності, що отримані через зв’язки *ISA*.

Зі складних типів, конструктор нечіткого типу (або нечіткого набору) визначає набір значень з заданої визначеної області типу (тобто з атрибуту, якщо користуватися термінологією реляційної моделі). Таким чином, справжніми екземплярами є лише та підмножина значень, яка відповідає об’єкту цього типу. Тоді кожен екземпляр, який взято з множини належить до цієї множини в межах від 0 до 1. Це також стосується й подібності елементів між собою. Тобто цей конструктор створює екземпляр у вигляді нечіткого набору, і в цьому наборі, відповідно, зв’язки між елементами теж нечіткі.

Конструктор неповного типу визначає конкретне значення із заданого діапазону. Цей конструктор має семантику «*XOR*», і це відрізняє його від конструктора нечіткого типу, де можливе знаходження декількох екземплярів.

Конструктор нульового типу використовується для представлення атрибутів, область яких спеціально розширено для можливості обробки різних типів нульових

значень. В моделі *ExIFO2* використовуються три таких: невідомий (*unk*), неіснуючий (*dne*) та безінформаційний (*ni*). Залежно від ситуації призначається різний нульовий тип. Наприклад, якщо ми чекаємо на автобус, але в даний момент його нема, але ми знаємо що він існує, то це невідомий тип. Якщо автобус приїхав, але ми не знаємо, чи це саме той, який нам потрібен, це безінформаційний тип. Якщо ж нам нема сенсу чекати автобус, бо потрібного для нас не існує на даний момент взагалі, то це неіснуючий тип.

Конструктор агрегації відповідає за з'єднання підтипу (частини об'єкта) з частиною, яка умовно являє собою весь інший об'єкт. В результаті створюється об'єкт вищого рівня. Цей конструктор отримує лише ті об'єкти, які підтримуються типом та є впорядкованими.

Конструктор композиції працює так само, які і агрегації, але гарантує індивідуальну структуру об'єкта, отриманого в результаті.

Конструктор групування це абстракція, в якій зв'язок між елементами розглядається як набір об'єктів. Семантично даний конструктор має логіку «*AND*», і на відміну від агрегації, усі об'єкти в групуванні є невпорядкованими. Але при цьому об'єкти мають той же тип. Фіксуються атрибути, які є багатозначними та чіткими. До прикладу можна навести перелік імен на заброньовані місця в театрі. Ці імена перераховуються через «І», всім їм належать власні місця, тобто перелік не є підмножиною.

Конструктор колекції працює так само, як групування, але при тому включає обмеження ексклюзивності для результуючих об'єктів.

Конструктор альтернативи обробляє структурно різні типи через зв'язок узагальнення та через посилене обмеження диз'юнкції між узагальненими типами.

Крім того, розглядаються також поняття узагальнення та спеціалізації. Узагальнення охоплює кілька сутностей і поєднує їх в узагальнену сутність. Спеціалізація ж охоплює єдину сутність, розбиту на кілька суб-сутностей.

ExIFO2 як концептуальна модель бере до уваги три види невизначеності. Це нечіткість, неповність та невідомість (або *null*). Саме таким чином *ExIFO2* може допомогти розробити систему на концептуальному рівні з врахуванням фактору

невизначеності. Як можна було побачити, саме для цього було сформовано три додаткових конструктора невизначеності (нечіткий, неповний, нульовий). Користуючись наведеними конструкторами можна явно представити атрибути, які мають невизначені значення. Це зроблено для розгляду невизначеності на рівнях атрибутів та об'єктах класу. Другим рівнем невизначеності є нечіткість екземплярів у наборах (клас/об'єкт) та нечіткість підкласів у суперкласах (підклас/клас). По-суті, подібні об'єкти збираються в групи для утворення класу.

Щодо атрибутів, вони можуть мати набір значень, тобто можуть бути багатозначними атрибутами. Ці значення можуть бути пов'язаними логічними операторами *AND, OR* або *XOR.*

Рівень об'єкт/клас представляє ступінь належності об'єкта до класу. Основна відмінність чітких класів від нечітких полягає в неточності меж останніх класів. Таким чином, одні об'єкти нечіткого класу можуть повноправно належати до цього класу (ступінь приналежності завжди дорівнює 1), а інші лише з деяким ступенем в межах від 0 до 1.

2.1.3. Проєктування геоінформаційної системи

Зазначивши теоретичні засади та коротко їх описавши, можна приступати до проєктування самої ГІС. Можна допустити, що основне джерело інформації, яке може надати програма такого напрямку використання – цифрова мапа. Нехай для початку мапа буде передбачати роботу з даними в *2D*-просторі, третій вимір при відображенні та аналізі мапи можна буде додати при майбутніх оновленнях за бажання та можливості. Отже, *2D*-мапа має відображати дані про певну місцевість чи територію, а така місцевість чи територія має назву. Це буде першим атрибутом. Також, мапа бере за основу зображення місцевості в певному початковому масштабі, це буде другим атрибутом. І зрозуміло, що ці обидва атрибути належать одному абстрактному типу – мапі.

Крім того варто пам'ятати, що мапа зазвичай також містить інформацію про низку географічний об'єктів. Так як поняття «географічний об'єкт» доволі

обширне, а крім того буде дуже часто використовуватися при експлуатації ГІС, варто звернути окрему увагу на його прилаштування при концептуальному моделюванні. Нехай для використання в системі поняттю «географічний об’єкт» буде присвоєно ім’я *Gobject*. Це скорочено з повної назви «*geographical object*» для зручності. Отже, географічні об’єкти, іншими словами деяка множина *Gobject*-ів має бути за своїм вмістом взаємовиключна, так як дуже бажано мати раніше встановлені рамки унікальності для об’єктів, особливо чи не основних за своїм призначенням в ГІС.

Gobject може визначати різні об’єкти, як річка, місто, будівля, вулиця, зупинка, тощо. Так як таких об’єктів потенційно може бути нанесено на мапу велику кількість, варто подумати про окрему ланку впорядкованості множини об’єктів типу *Gobject*. За своїм призначенням для такої ролі підходить список об’єктів, який допомагає їх групувати, а на концептуальному рівні для цього варто застосувати конструктор колекції, який, нехай, зветься *Gobjects*, який за своєю назвою буде вже вказувати на множину відповідних об’єктів.

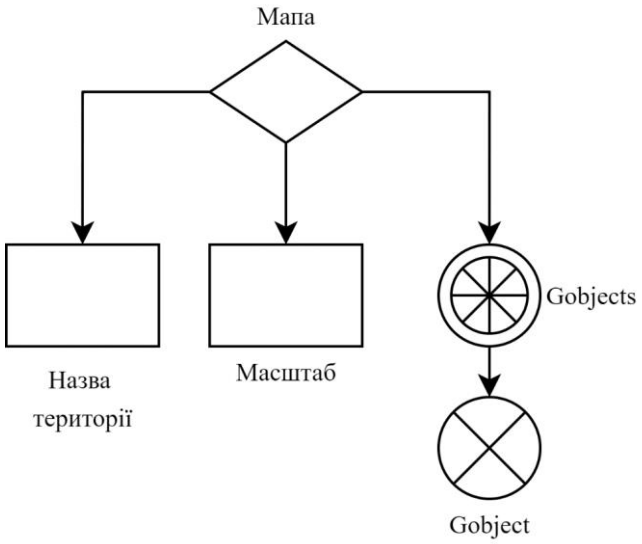

Рис. 2.4. Основний абстрактний тип «Мапа» з віткою географічного об’єкту *Gobject*

Для представлення *Gobject* було обрано агрегацію, так як даний тип, в свою чергу, формують певні властивості, що природньо підходить під поняття агрегації

(агрегування декількох властивостей для формування типу через відповідний конструктор).

Географічний об’єкт в ГІС даного проєкту може мати такі властивості:

– інформаційний опис;

– геодані;

– дані графічного відображення.

Перша властивість відповідає за дані, які не обчислюються, а виводяться лише як описова інформація географічного об’єкту, яка не залежить від геоданих. До прикладу, маємо пересувний шпиталь, нехай у вигляді мобільного пункту швидкої допомоги на базі спеціально обладнаного автомобіля. Знаючи основні дані про даний об’єкт, такі як марка, номер, максимальна чи середня загальна швидкість, графік роботи чи список персоналу пункту, користувач ГІС може отримати коротку справку або розгорнуту інформацію про певний *Gobject* (в залежності від об’ємів даних інформаційного опису та потрібності їх надання в ГІС), але ці дані не є геоданими, вони є набагато більш сталими та потребують зазвичай куди менше циклів оновлення інформації до стану актуальної, крім того, інформаційний опис може дублювати або використовувати початкові дані, призначені для подальшої обробки та різних комбінацій з геоданими, але «в ходу» знаходяться саме дані трекінгу, що являють собою ті ж геодані.

Геодані, або ж дані трекінгу – це важлива група інформації, яка є основою для роботи ГІС. Мова йде про трекінг стану географічного об’єкту.

Трекінг стану географічного об’єкту є в певній мірі більш загальним визначенням і включає в себе відслідковування географічних координат об’єкта та різних з цим пов’язаних параметрах (переміщення об’єктів, швидкість пересування, кількість і тривалість зупинок, можливість зберігання маршруту) а також за допомогою різних датчиків (для автомобілів це, наприклад, температура двигуна, показання тахометра, знос гуми, необхідність заміни масла тощо). Детальніше зупинимося на відслідковуванні положення об’єкта в просторі через *GPS* (*Global Positioning System*).

GPS-трекінг призначений для опитування джерела з потрібною інформацією (*GPS*-трекера на самому об’єкті, сервера, супутника чи іншого джерела з потрібною інформацією) про просторове положення об’єкта на місцевості. Частота опитування може варіюватися, це залежить від декількох факторів: наскільки часто об’єкт міняє своє положення, наскільки потужна апаратура та канал зв’язку, щоб ці опитування здійснювалися з певною частотою, та наскільки такі дані взагалі потрібні для вираховування коректних результатів за вимогою. Загалом, якщо маємо *Gobject*, який представляє певний будинок, гору, пам’ятник чи щось на кшталт подібного зазвичай нерухомого роду об’єкти, то опитування щодо їх положення, напевно, все ж не слід оновлювати занадто часто. Інша справа, якщо такий трекінг стосується чогось рухомого, будь то автомобіль, літак, чи дика тварина з *GPS*-трекером. В такому разі вже слід задуматись про більшу частоту опитування координат, особливо якщо маємо справу з системами обробки даних в реальному часі.

Варто зазначити, що способи отримання даних трекінгу розглядаються для загального розуміння таких шляхів та перспективи можливості їх впровадження до роботи ГІС-застосунку, але все ж для початку система в проєкті буде обробляти вже задані дані трекінгу, так як предметом дослідження є робота самої системи, а не попередніх ланок добування даних для обробки, які можуть потенційно надати різні джерела – як у вигляді тестових даних, так (в перспективі) і фактичних з застосуванням більш розвинутого обладнання.

Дані графічного відображення вже належать безпосередньо до роботи ГІС, особливо до пункту візуалізації. По-суті, дві попередніх властивості *Gobject* потрібно представити в графічному форматі (точніше, лише потрібні дані з цих властивостей для кращого концентрування користувача на результатах роботи ГІС, але все ж дані для візуалізації – річ дуже відносна і залежить від ситуації та вимог).

Тип інформаційного опису, загалом, є необов’язковим, так як просторової інформації зазвичай достатньо для загального базового опису точки на карті. Але при тому, поскільки ГІС розрахована на зручний інтерфейс, варто все ж мати такий тип в структурі програми, так як з себе він не надто складний, а додаткова

інформація при виведенні даних ніколи не завадить, або, принаймні, така можливість, яка закладена під час проєктування ГІС.

Тип «інформаційний опис» має в своєму складі тип під назвою «*Aspatial*» (апросторовий, непросторовий). Проміжною ланкою між цими типами буде тип *Aspatials* у вигляді конструктора групування. Саме поняття «непросторовість» використовуються при побудові типу через те, що сам тип є необов'язковим, і може бути не використаний, тобто опису такого виду може в певних випадках не існувати. Конструктор групування для представлення непросторовості використаний через те, що об'єкт може мати кілька інформаційних описів. Тип *Aspatial* являє собою групу, до якої можуть належати деякі атрибути (площа, населення, назва тощо), на діаграмі це вільний тип. Але сама ця група повинна мати власне ім'я.

Непросторові типи зазвичай породжують мінімум три підтипи, а саме:

– *StrAtr*;

– *IntAtr*;

– *FuzzAtr*.

StrAtr – це атрибут, який має на меті опис значення в вигляді рядків, тобто буквенним варіантом, найближчим до представлень людини. Прикладом застосування є назви різноманітних об'єктів чи їх типів, груп, до яких ті належать.

IntAtr – це атрибут, що здійснює опис в числовому вигляді. Застосовується вельми часто, якщо потрібно точно виразити кількість, номерний порядок чи інше значення, пов'язане з лічбою.

FuzzAtr – атрибут, що відповідає за нечітке визначення певного поняття. Теоретично, він може розширити інформацію з *IntAtr*, але доведеться зіткнутися з діапазоном оброблюваного поняття. Наприклад, перше що спадає на думку – населення на певній території. Якщо територія є відносно великою, а люди можуть її вільно покидати чи повертатись (не говорячи вже про народжуваність чи смертність), то такі дані вельми часто змінюються, і тим же атрибутом *IntAtr* буде описувати такі дані незручно занадто часте оновлення. Можливо, в деяких системах моніторингу, такі дані й потрібні в точній мірі, і їх дійсно варто залишити

у вигляді *IntAtr*, але все ж, якщо на цей аспект кількості не робиться великий акцент при розрахунках, то краще буде представити такі дані у вигляді діапазону значень, тобто через *FuzzAtr*.

Географічний об’єкт має певні геодані, просторові властивості, так як мапи й розраховані головним чином на те, щоб в найпростіший спосіб (через візуальне представлення в деякому масштабі й відносне розташування таких об’єктів певної зони) їх представляти. Для цього потрібно відтворити визначену геометрію, яка може включати в собі різні візуальні примітиви (точки, лінії, області тощо). Повертаючись до абстрактного типу, який являє собою географічний об’єкт в даній ГІС (*Gobject*), то саме він забезпечує даними геометричні структури для обробки та відтворення на цифровій мапі. Через концептуальну модель *ExIFO2* такі просторові властивості представляє тип колекції «Геодані», якому надано відповідальність за обробку всіх просторових властивостей об’єкта. Конструктор колекції тут використовується тому, що об’єкти, які мають входити в цей тип, повинні бути взаємовиключними. Тип «Геодані», виражений через конструктор колекції, має в своєму складі об’єкти *Geo,* які є екземплярами типу «Геодані».

Підтип, які має в своєму складі об’єкт *Geo* це точка точка, лінія та область. Точка є смисловою основою лінії та області (множина точок в тому чи іншому вигляді), але в даному випадку ієрархічну структуру не представляє. Це зумовлено тим, що точка в першу чергу являє собою одну зі складових просторових властивостей, й є рівноправним поняттям у цьому сенсі щодо ліній та областей, так як для цих складових природньо існувати разом. Наприклад, потрібно описати вулицю на мапі для відтворення. Вулиця – це будинки та дорога, а поскільки маємо справу з трекінгом, то явно мають бути географічні об’єкти, що мають власне положення в просторі, тобто конкретну точку знаходження на мапі. Будинки представляються як малі області, дорога на вулиці (можливий простір для маршруту) – як лінія, а пункт призначення – як точка. Таким чином в сенсі побудови мапи точка, лінія та область – рівноправні об’єкти, а з точки зору геометрії – ієрархічні.

Точка представляє геометричний аспект об’єкта, для якого має значення лише положення в просторі по осям координат (або об’єкт підходить по масштабу відносно інших об’єктів на мапі як точка, або береться умовний геометричний центр об’єкта чи потрібна для його точкового позначення частина). Зрозуміло, що для об’єкта представленого точкою не вживаються характеристики площі чи протяжності, так як для цього існують підтипи лінії та області відповідно.

Підтип «Точка» має атрибут ідентифікатора (*ID* точки), щоб краще контролювати існування такого об’єкта. Крім того, для групового представлення значень двовимірних координат (*X* та *Y*) використовуються агрегація, а отже присутня проміжна ланка конструктора агрегації під назвою «Координати».

«Лінія» – це абстракція для об’єктів, які мають характерну властивість – протяжність. Цей підтип гарно підходить для відображення таких об’єктів, як-от дороги, річки, огорожі, лінії електропередач та подібні об’єкти з протяжністю. Як можна здогадатись, лінія структурно пов’язана з підтипом точки, але не через просту множину точок, які складають всю лінію, а через дві основні точки – «Початок» та «Кінець».

Третій підтип в складі *Geo* це «Область». Область має позначати площу, обмежену лініями, що належить до певної однорідної або близької до однорідної множини точок з подібними характеристиками. Найчастіше такі області створюються як представлення чогось однорідного на площі, при тому в різних представленнях: це може як вигляд зверху будинку чи іншої перешкоди, щоб мати представлення про обмеження здійснення руху; це також може бути якась абстрактна зона, наприклад, скупчення виду тварин чи поширення визначеної рослинності на території, що представлено виключно для більш наочної подачі даних на мапі.

Дані графічного відображення відповідальні за програмну обробку та вивід даних в графічному вигляді для користувача. Неможна з впевненістю сказати, яка частина під час розробки програм завжди належить розробці чи проектуванню графічного інтерфейсу користувача (*GUI – graphical user interface*). Певні програми потребують такого інтерфейсу, деякі ні, але якщо говорити саме про випадки

необхідності частої взаємодії користувача та програми, то виникає необхідність не просто категоризації та розумного розміщення даних на приладі відображення (монітор, голографічна панель, тощо), а просто інтуїтивно зрозумілого для способу мислення та сприйняття людини способі подачі необхідної в певний момент часу інформації. Недарма за останні десятиліття набув популярності віконний інтерфейс (що є досягненням та, як мінімум, заслугою популяризації операційної системи *Windows*), та матеріального дизайну при створенні графічних інтерфейсів, який розповсюджувався з побудови веб-сайтів, потім це підхопили мобільні ОС, а зараз навіть такий дизайн впроваджується в нових ревізіях *Windows*.

Отже, слід на етапі проектування визначити, наскільки складною за вимогами до кінцевого продукту має бути застосунок, адже складність має також зворотню сторону – контроль користувача над процесом обробки даних. А це напряму впливає й на побудову інтерфейсу.

Знаючи важливість присутності графічного інтерфейсу для таких програм, де ведеться інтенсивний обмін візуальними даними з користувачем (а ГІС якраз відноситься до застосунків, де візуальна частина програми дуже важлива для злагодженої роботи користувача з системою), повернімося до проєктування підтипу «Дані графічного відображення». Суть в тому, що хоч з часом й візуальні елементи на сучасних системах еволюціонували до згаданих вже віконних інтерфейсів та матеріального дизайну, але текст представляє, за деякими винятками, основний пласт інформації з яким користувач працює чи не найбільше навіть в графічно-орієнтованих програмах. Таким чином текстовий вміст під назвою «Текст» має бути в першу чергу бути включено до графічних властивостей («Дані графічного відображення») типу *Gobject*.

Разом з текстом можливо обрати й спосіб його відображення. Для цього підійде графічна властивість «Шрифт». Не скрізь є нагальна потреба у використанні декількох шрифтів для відображення тексту, але враховуючи, що ГІС це все ж в тому числі й інформаційна система, потрібно врахувати можливість великої кількості тексту при виведенні користувачеві, а отже, може й з’явитися потреба й для розрізнення символьних рядків, наприклад, заголовок чи назва в

інформації по мапі може відображатися одним текстом, а інформація про певний географічний об’єкт – іншою. Так це додасть потрібну різноманітність до подачі інформації, спростивши її сприйняття.

Інша графічна властивість це «Текстура». Це фактично є графічним файлом, картинкою, яка в той чи інший спосіб відображається на екрані. В ГІС на мапі це фактично вся інформація, крім тексту – зображення місцевості, піктограми, умовні позначення, навіть віртуальні фонові панелі, на які накладається текст – все це зображення, які в програмі підвантажуються з графічних файлів. Хоча текстури виконують іншу роль ніж текст, але є так само важливими в графічному інтерфейсі користувача.

Графічна властивість «Колір» відповідає за забарвлення двох основних елементів: «Текст» та «Текстура». Для тексту використовується зміна забарвлення шрифту, для текстури – або накладення відповідного кольорового фільтру, або завантаження текстури з початково вказаним кольором, якщо така в наявності.

Ще одна графічна властивість, яка може знадобитися для більш комфортного споживання інформації – «Розмір». На відміну від положення графічних елементів на екрані, які зазвичай являють собою дві координати в *2D*-просторі *X* та *Y*, які мають прив’язку до ширини й висоти в пікселях дисплейної частини пристрою візуального відтворення, властивість «Розмір» впливає на наявну ширину й висоту тексту чи текстури, пропорційно збільшуючи чи зменшуючи такі параметри. Знову ж, це зумовлено різноманітністю подачі інформації через інтерфейс, так як такий елемент, як піктограма якогось пункту меню не повинна займати багато місця, а навпаки – економити його, й часто вживається в панелях меню, де текст використовує занадто багато місця відносно інших графічних елементів. Але в тому ж тексті також можуть бути присутні певні місця, де його розміри варто збільшити чи зменшити відносно розмірів звичайного використання. Найкращий приклад – заголовки. Крім чисто візуальної користі розрізнення інформації існує також й нагальна, яка пов’язана з різною роздільною здатністю дисплеїв, з якими працює користувач, і на які виводиться інформація через графічний інтерфейс. Для цього

потрібно розраховувати розміри графічних елементів на екрані з різними роздільними здатностями, а також масштабувати *GUI* в цілому.

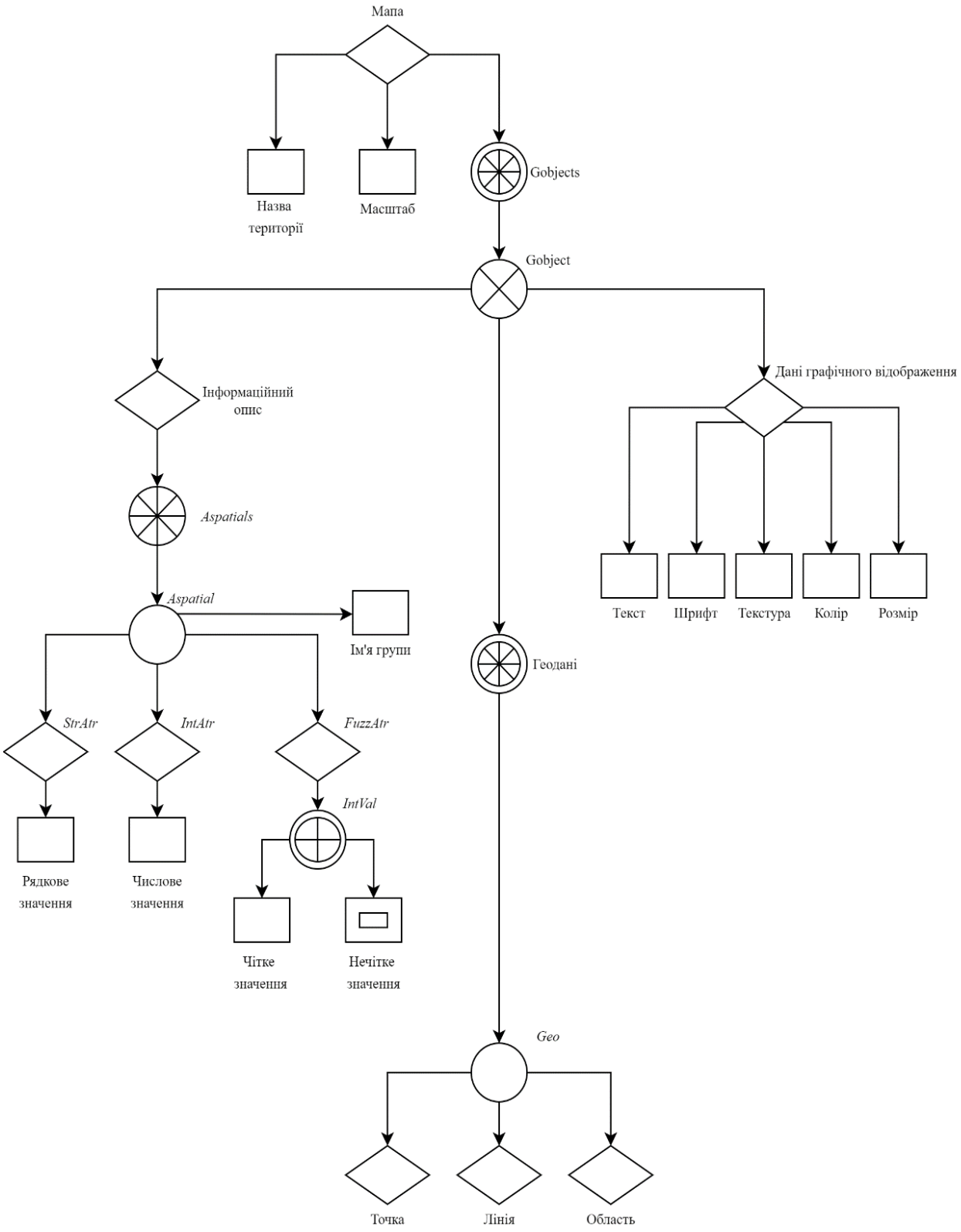

Рис. 2.5. Загальний вигляд графу по ExIFO2 для геоінформаційної системи

2.2. Алгоритми аналізу даних трекінгу в геоінформаційній системі

2.2.1. Дані трекінгу джерел поповнення енергії

Основне завдання трекінгу – контроль за переміщенням і станом об’єктів [8]. Переміщення об’єктів представляється як координати місцезнаходження об’єкту в просторі, а стан об’єкту – це показання з його датчиків та/або персоналізовані дані, які допомагають класифікувати та вирізнити об’єкт серед інших.

Основними типами географічного об’єкта в даній ГІС є джерело поповнення енергії та споживач. Саме характеристики таких об’єктів, що входять до даних трекінгу взятих з зовнішніх інформаційних джерел входять до аналізу та подальшої візуалізації через різноманітні позначки на цифровій мапі ГІС. До даних, за якими здійснюється трекінг джерел поповнення енергії (далі ДПЕ) входять не лише просторова інформація, як довгота широта місцезнаходження географічного об’єкту, але також й такі характеристики, як тип конектора чи доступність ДПЕ. Загалом дані трекінгу мають такий перелік:

– місцезнаходження джерела;

– працездатність джерела;

– тип коннектора джерела;

– довжина шляху до джерела.

Місцезнаходження джерела потрібно для орієнтування авто відносно положення таких станцій зарядки, тобто визначення довжини переміщення до таких станцій, і, звісно, для візуалізації цих географічних об’єктів на цифровій мапі згідно наданих даних трекінгу.

Працездатність джерела потрібно перевірити для того, щоб усунути проблеми, які можуть статися незалежно до номіналів чи положення станції – наприклад, недієздатність певної станції на момент здійснення аналізу через поломку чи проведення профілактичних робіт. В такому разі по характеристикам зарядна станція підходить для авто, але не може бути використана через інші причини, а отже, немає сенсу в її подальшому порівнянні.

Тип конектора джерела потрібен для визначення того, чи станція може бути використана для живлення авто через стандартні роз'єми. Крім того, є певні типи роз'ємів, які можуть використані для декількох типів зарядних станцій, що дає більше можливостей у пошуках станції-кандидатів, але також ускладнює порівняльний аналіз.

Довжина шляху потрібна лише як додатковий, хоча й важливий критерій, який надається як дані трекінгу, тобто для того, щоб на останньому етапі аналізу станцій-кандидатів визначити, шлях до якої є мінімальним відносно інших станцій.

Аналіз даних трекінгу ДПЕ діє разом з даними про споживача, так як для того, щоб порівняти джерела між собою, потрібно орієнтуватися на відповідні дані споживача енергії, так як в даному разі не можна вибрати між декількома географічними об'єктами без узгодження їх характеристик не лише між собою, а й відповідно до характеристик живлення електрокару, який виступає в ГІС споживачем. Отже, ГІС повинна мати доступ до таких даних, як ємність акумулятора електрокару, поточний процент розряду та деякі дані, на кшталт середньої швидкості авто, його положення на мапі та час розряду повного акумулятора при середньому споживанню енергії, що надається як зовнішні дані, якими оперує ГІС в поєднанні з даними трекінгу джерел.

2.2.2. Виявлення джерел поповнення енергії в зоні досяжності споживача

Застосовано швидкий алгоритм розрахунку з використанням локальних даних. Нехай це буде певний район міста чи невелике СМТ (селище міського типу) потрібно розділити на сітку територіальних квадратів. Кожен такий квадрат мапи матиме власний ідентифікатор, в тому числі й конкретні дані про свої координати та географічні об'єкти (*gobject*-и в системі), що знаходяться в межах території таких квадратів. Наведеною сіткою можна розбити відносно велику територію на індивідуальні квадрати на основі довготи й широти, інформація про які зберігається в пам'яті.

Також для того, щоб полегшити системі розрахунки та зменшити початковий список географічних точок для аналізу, слід встановити певну область пошуку. В межах тестування це може бути окрема мапа території, всі наявні станції на якій підлягають перевірці критеріям відповідності, для чого здійснюватиметься порівняльний аналіз. Але є й інший спосіб – пріоритетне орієнтування не на межі місцевості на мапі, а на положення авто. Таким чином, формується область пошуку в межах так званої «зони досяжності» авто, яка представляє собою круглу зону з центром в точці положення авто на мапі.

Слід дотримуватися наступних кроків, щоб визначити географічні об’єкти, які потрібно внести до переліку як такі, що знаходяться в зоні досяжності електромобіля (або зазначити про відсутність таких об’єктів):

1) Знайти час розряду акумулятора з поточного відсотку до нуля (від часу розряду акумулятора від 100% до 0 за середнього використання відняти час, який пройшов від використання авто з моменту повної зарядки).

2) Знайти яку довжину шляху зможе подолати авто з поточним відсотком заряду (за загальною середньою швидкістю та часом, який потрібен для розряду акумулятора з поточного відсотку до нуля).

3) Визначається той квадрат мапи по ідентифікатору, де розташована відправна точка (електрокар) відповідно до значень широти й довготи. Такий квадрат називається центральним.

4) По черзі визначається відстань *AB* між станцією (точка *B*), що знаходиться в квадраті відносно точки на мапі, де знаходиться авто (точка *A*) по прямій лінії. Якщо твердження $AB <= L_{макс.шляху}$ (де $L_{макс.шляху}$ – це довжина шляху, яку може подолати авто з поточним відсотком заряду до повного розряду акумулятора) справджується, такий об’єкт-кандидат вноситься до переліку, в якому надалі будуть порівнюватися його характеристики, так як він знаходиться в зоні досяжності авто.

5) Обираються за ідентифікаторами найближчі квадрати сітки до центрального квадрату з відправною точкою (знайдений на кроці 3) таким чином, щоб обрані квадрати оточували попередню зону квадратів. На першій ітерації

пошуку оточуючих квадратів попередню квадратну зону складає лише один центральний квадрат. Оточуючі квадрати допомагають знайти перелік географічних точок в зоні досяжності авто. Така зона що має форму круга з центром в точці місцезнаходження авто. Радіусом круглої зони досяжності є довжина шляху $L_{\text{макс.шляху}}$.

6) Кроки 4-5 повторюються включно до тих пір, поки така квадратна зона з оточуючих квадратів повністю не покриває уявний круг зони досяжності по завантаженій мапі. В результаті формується перелік точок джерел поповнення енергії, які знаходяться в зоні досяжності електромобіля.

2.2.3. Відбір джерел поповнення енергії в зоні досяжності споживача

Не кожне джерело поповнення енергії у вигляді станції зарядки може задовольнити потреби електромобіля. Хоча технічні характеристики для кожного стандарту зарядки заздалегідь встановлені, все ж єдиного стандарту роз'єму (а як наслідок і конектору зарядки) поки не встановлено. Якщо брати за основу класифікацію зарядних станцій для електромобілів регіону Європи, тоді наразі зафіксовані такі характеристики стандартів [9]:

Mode 1 – тип зарядки, який може бути здійснений навіть від побутової електромережі. Наразі практично не надається виробниками як можливість для підзарядки електромобілів, так як має дуже низьку ступінь безпеки зарядки як для оператора, так і для самого автомобіля. Здійснюється без спеціального обладнання, потрібна лише стандартна розетка й адаптер змінного струму з сумісним конетором. Має низький часовий інтервал для підзарядки – 12 годин в середньому. Не буде розглядатися в ГІС.

– змінний струм: до 16 А;

– напруга: 220–240 В;

– потужність: 2–4 кВт·ч.

Mode 2 – стандартний тип зарядної станції на змінному струмі, що вже обладнаний власною системою захисту всередині кабелю. Найчастіше

зустрічається на станціях підзарядки та часовий інтервал поповнення енергії авто десь 6-8 годин.

– змінний струм: до 32 А;

– напруга: 220–240 В;

– потужність: до 7–8 кВт·ч.

Mode 3 – найпотужніший тип, який використовують станції на основі змінного струму, що може зарядити електромобіль в середньому за 3-4 години.

– змінний струм: до 63 А;

– напруга: 220–230 В;

– потужність: до 43 кВт·ч.

Mode 4 – зарядні станції, що використовують вже постійний струм, що забезпечує заряджання акумулятора від 0% до 80% за 30 хвилин. Ціна встановлення такої станції доволі висока, на що впливає також і потреба проведення зазвичай окремої лінії електропостачання. Це пояснює рідкість таких станцій, особливо в Україні.

– постійний струм: до 400 А;

– напруга: до 600 В;

– потужність: до 250 кВт·ч.

Перші три типи можна назвати «повільними зарядками», вони будуть заряджати самий звичайний електрокар годинами, а ось наступні зазвичай здатні зарядити електрокар за приблизно за півтори години а то й менше, тому їх можна вважати «швидкими зарядками».

Також, кроки алгоритму для перевірки технічних характеристик джерел не можна зробити без розуміння типів роз’ємів зарядки на електрокарах. В переліку представлені найбільш вживані на ринку електрокарів, так як через відсутність єдиного стандарту типів ще більше.

Type 1 (J1772) – п’ятиконтактний роз’єм сумісний зі стандартами станцій *Mode 2* та *Mode 3*. Створений та використовується для регіонів США та Японії.

Рис. 2.6. Приклад схематичного зображення роз’єму (*Type 1 J1772*, США/Японія)

Type 2 (Mennekes) – семиконтактний роз’єм також сумісний зі стандартами станцій *Mode 2* та *Mode 3*, але створений та використовується в європейському регіоні.

CHAdeMO – (*charge de move*, означає французькою «зарядись для руху») двоконтактний роз’єм постійного струму від компанії *TEPCO*, розроблений разом з участю великих японських автовиробників. Розповсюджений у японських, американських та ряду європейських електромобілів. Працює зі станціями, що розраховані на потужний режим зарядки *Mode 4.*

CCS Combo – (*combined charging system*, комбінована система зарядки) комбінований тип роз’єму, основною перевагою над іншими є можливість використання як звичайні режими зарядки (*Mode 2, Mode 3*), так і швидкий *Mode 4.* Крім того, такий комбінований тип гарантує безпечне використання одного з двох роз’ємів звичайної зарядки - *Type 1* при регіоні авто США та Японії, а *Type 2* – при регіоні Європи. Відповідно до сумісності, існує два окремих роз’єми: *Combo 1* та *Combo 2*.

GB/T – (*Guobiao tuījiàn,* перекладається як «рекомендований національний стандарт») тип роз’єму, який розроблено виключно для китайського ринку. Основна проблема цього стандарту – несумісність з вищезгаданими роз’ємами по технічним й електронним частинам. По даному стандарту, виробники роблять окремі розетки й, відповідно, конектори для змінного струму звичайної зарядки й постійного струму швидкої зарядки.

Роз’єм *Tesla* – ексклюзив для електромобілів відповідної компанії. Такий роз’єм дає можливість використовувати надшвидку фірмову зарядку *Tesla Supercharger*. Але навіть цей роз’єм ділиться на два підтипи – для США та Європи.

Знаючи про режими зарядок й типи роз’ємів, можна сформувати такі кроки алгоритму підбору станції для автомобіля:

1) Перевірка статусу працездатності станції, так як нема сенсу робити перевірку тієї зарядної станції по характеристикам, якщо вона не працює.

2) Зіставлення типу роз'єму конектора на станції з роз'ємом вибраного електрокару.

3) Якщо тип роз’єму авто *CCS Combo*, тоді відбувається додаткова перевірка на наявність станцій з сумісними конекторами з *Type 1* та *Type 2*.

4) Якщо після кроку 3 залишається більше однієї станції-кандидату, тоді обирається та, до якої довжина шляху відносно споживача найменша.

2.3. Вибір компонентів для розробки геоінформаційної системи

Проєктна ГІС має зберігати, аналізувати та візуалізувати дані вибору джерела поповнення енергії. Для того, щоб система могла бути розроблена хоча б для етапу тестування її основного призначення, потрібно застосувати такі програмні компоненти ГІС [10]:

– пакети для аналізу даних ГІС;

– статистичні пакети;

– пакети візуалізації даних.

Це основна частина ГІС. Пакети для аналізу даних потрібні для різного типу обробки інформації, яка допомагає розв’язати задачі системи. Такі пакети будуть розроблені окремо у вигляді програмних функцій, що виконують підбір найкращої по різним критеріям зарядної станції на мапі для авто.

Статистичні пакети потрібні для перевірки системи без зосередження на добуванні та передачі даних, і застосовуються як зовнішні дані для обробки, на основі яких здійснюють подальші розрахунки за алгоритмами ГІС. Так як ГІС

розробляється з перспективою співпраці з *GPS*-сервісами, які будуть надавати дані про координати географічних об’єктів та маршрути між двома точками, тоді така інформація має або дійсно надаватися при потребі, або зберігатися у вигляді вже добутих даних. Другий варіант більше підходить для розробки перших версій системи, так як потрібно впевнитись, що ГІС коректно обробляє надані перевірені заздалегідь за значенням дані, для яких результати або вже передбачені іншими системами, теоретично, або які легко перевірити безпосередньо при тестуванні проєктної системи. Потім збільшити варіативність підготовлених наборів, і тільки при правильній роботі системи з попередніми вже збереженими наборами надавати актуальні вхідні дані з *GPS*-сервісів. Це стосується як даних трекінгу джерел поповнення енергії, так і дані про електрокар та стандарти, потрібні програмі для аналізу. Таким чином, розробник зосереджується на правильній роботі системи та коректних її результатах.

Пакети візуалізації даних потрібні для економії часу на розробку програмних засобів графічного відображення. Якщо пакети для аналізу даних для даного проєкту розробляються самотужки, а статистичні пакети даних це лише набір згенерованих або добутих раніше значень, який залежно від потреб системи чи її тестування може бути як впорядкованим так і ні, пакет візуалізації в даному випадку буде використовуватися сторонній. Тож, потрібно зупинитися на цьому детальніше.

Як пакет візуалізації можна розглянути програмну бібліотеку *SFML* (*Simple and Fast Multimedia Library*) [11]. Це відома бібліотека з відкритим програмним кодом, яка має в своєму складі чимало засобів для надання тому чи іншому проєкту можливостей справжнього мультимедійного додатку, тобто присутня робота зі звуковими та візуальними ефектами, системними ресурсами та вікнами графічного інтерфейсу. Так, сама бібліотека має в своєму складі чотири відповідних модуля:

– *Graphics* (відображення графічних ефектів та примітивів);

– *Audio* (робота зі звуковими ефектами);

– *System* (багатопоточність, робота з програмним часом);

– *Window* (робота з графічними вікнами).

Такий набір модулів дозволить зосередитись розробнику на створенні та покращені алгоритмів призначення самої системи, а не розробляти власні звукові чи графічні бібліотеки, що є суттєвою економією часу та сил. *SFML* – кросплатформна бібліотека, а отже є можливість створювати програми з мультимедійними засобами взаємодії з користувачем на таких настільних системах, як *Windows*, *MacOS* та *Linux*, автори бібліотеки також анонсували розробку даної мультимедійної бібліотеки на мобільні системи, як *Android* та *iOS*.

Дана бібліотека вирізняється легкістю користування та малими системними вимогами до обробки її модулів. Особливо це помітно при створенні програм з *2D*-візуалізацією, так як основні візуальні можливості графічного модуля *SFML* розраховані саме на відображення двовимірної графіки на основі відповідних геометричних примітивів. Для *3D*-відображення графіки, наприклад для мап з ландшафтом ця бібліотека не підійде, доведеться частково (в парі з *SFML*) або повністю використовувати іншу бібліотеку з відкритим вихідним кодом – *OpenGL*. Але так як для початку проєктну ГІС потрібно розробити та протестувати спершу з базовим варіантом відображення без додаткових можливостей (вони можуть бути додані пізніше), тобто спочатку зробити повністю робочі аналіз та візуалізацію в двовимірному просторі, а потім нарощувати складність. Отже, на даному етапі *SFML* буде достатньо. Крім того, бібліотека має вільну модель розповсюдження навіть для комерційних проєктів.

Для відображення двовимірної графіки за допомогою даної бібліотеки використовується бібліотечний модуль *Graphics*, що забезпечує спрайтову анімацію, а в випадку відображення ресурсноємної графіки, до якої відноситься мапа місцевості чи картографічні знімки, можливо застосувати фрагментарний показ, що зекономить системні ресурси. Основою для побудови геометрії в *SFML* є прямокутна система координат з двома осями.

Перевагами мультимедійної бібліотеки *SFML* є, однозначно, простота використання програмних інструментів, що надаються бібліотекою, велика кількість користувацьких інструкцій по написанню програм на основі *SFML* різної спрямованості, невисокі системні вимоги для користування бібліотекою,

можливість портативного користування (непотрібно інсталювати в систему, ли прописати шляхи до файлів), просте налаштування роботи з графікою за допомогою бібліотеки. До недоліків можна віднести відсутність стабільних версій бібліотеки для розробки програм на мобільні операційні системи, порівняно, відносно неофіційної, мала кількість описових документів від розробників та повільний процес публікації нових оновлень, в яких виправляються помилки, та додаються нові можливості, яких потребує спільнота користувачів даної бібліотеки.

2.4. Висновки до розділу

В другому розділі кваліфікаційної роботи було розглянуто теоретичну частину роботи над ГІС. Було прийнято рішення застосувати концептуальну модель для проєктування ГІС. Описано значення основні ідеї та застосування концептуальних моделей, та обрано конкретну модель з поміж декількох для побудови системи. Було здійснено проєктування за моделлю під назвою *ExIFO2*, яка зберігає сильні сторони семантичних підходів. Використання такої моделі в основі ГІС ще на етапі проєктування забезпечує системі перспективи додавання стійкості до невизначеності та нечіткості початкових даних, які оброблятиме система. Це новий підхід до концептуальної побудови, що забезпечить можливість покращення подання нечітких даних, які властиві для даних, які часто обробляють ГІС. Роз’яснено та наведено приклади представлень на основі такої моделі в графі.

Також, на основі *ExIFO2* була створена структура на основі графу основної частини ГІС-застосунку – цифрової мапи. Детально описано основний набір даних, який фігурує в ГІС – географічний об’єкт, названий *Gobject* в самій системі. Він складається з трьох основних типів: інформаційний опис, геодані та дані графічного відображення, які також описані та позначені на часткових графах.

Було визначено та описано основні два алгоритми, які виконують поставлені перед системою задачі відповідають за аналіз та візуалізацію даних трекінгу джерел поповнення енергії, а саме:

– відбір найближчих до споживача станцій, яких може досягти споживач;

– визначення джерел поповнення енергії з характеристиками, що відповідають запитам споживача.

Крім того, були визначені компоненти для розробки геоінформаційної системи.

Було розгорнуто описано можливості бібліотеки *SFML*, з яких модулів вона складається, її переваги й недоліки.

Отже як підсумок, для створення ГІС на етапі проєктування було застосовано сучасну модель *ExIFO2* з перспективою розвитку стійкості до нечіткості початкових даних, де граф може бути перетворений у логічні структури на етапі розробки, на основі яких неважко буде написати програмні інструкції та описати класи. ГІС має виконувати два основні алгоритми – обирання джерел поповнення енергії з характеристиками, що відповідають потребам елемента живлення споживача (наприклад, електромобіля) й візуальна подача маршруту до таких географічних об’єктів. Все це супроводжується візуалізацією у мінімалістичному стилі на основі мультимедійної бібліотеки *SFML*, що робить ГІС-застосунок невибагливим до потужності графічних компонентів пристрою користувача.

РОЗДІЛ 3
РОЗРОБКА ГЕОІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Вибір програмних засобів для розробки геоінформаційної системи

Отже, здійснивши опис програмних аналогів, проєктування ГІС та опис її основних алгоритмів аналізу даних трекінгу, слід вибрати компоненти для її розробки.

Перш за все варто зазначити, що оскільки проєктування здійснювалося на основі моделі даних *ExIFO2*, яка передбачає гарну сумісність моделі з об'єктно-орієнтованою філософією створення програмного коду, так як програма собою являє, фактично, множину об'єктів, що взаємодіють між собою, що гарно перекликається зі сформульованими раніше діаграмами концептуального створення ГІС. Це означає, що коло огляду можливих мов для розробки ГІС варто звузити до лише тих мов програмування, які належать до високорівневих та об'єктно-орієнтованих. Тож, можна розглянути об'єктно-орієнтовані мови програмування для створення ГІС:

– *Java*;

– *C#*;

– *C++*.

Java має багато прихильників серед розробників, насамперед, тих програмних продуктів, які можуть запускатися на різних апаратних платформах – від мобільних телефонів до банкоматів. Особливістю цієї мови відносно інших є вмонтований засіб для очистки пам'яті, таким чином одночасно звільняючи розробника від додаткового контролю над ресурсами пам'яті, але тим самим і робить чималі обмеження щодо більшої гнучкості та швидкодії програми в цілому.

C# – мова програмування, яка створювалася як мова створення продуктів компанії *Microsoft*. Синтаксис мови нагадує як *Java*, так і *C++*. Має також багато сфер застосування, наприклад офісні програми, служби *Windows*, мобільний софт

та навіть може застосовуватися як *web*-мова. Із мінусів можна назвати його легке дизасемблювання – тобто без додаткових заходів безпеки більшість коду зможе бути несанкціоновано прочитано, а це відкриває не тільки структуру програми конкурентам, а й загрожує загальній безпеці системи, написаній на цій мові. Крім того, використовуються принцип компіляції, який передбачає перетворення інструкцій в машинний код лише по мірі необхідності під час роботи застосунку. Це потенційно може знизити швидкість виконання програми навіть на потужних машинах.

C++ – найвідоміша мова об'єктно-орієнтованого програмування, яка дозволяє користувачеві без додаткових інструментів працювати з пам’яттю машини, тим самим збільшуючи гнучкість мови, але покладаючи відповідальність за будь-які подібні дії на розробника. Найкраще підходить для втримання балансу між високорівневістю та отримання контролю над системними ресурсами, що також відносно двох попередніх мов позитивно впливає на швидкість, підвищуючи складність створення коду. Такий баланс у суті цієї мови програмування присутній ще з 80-х років, коли дана мова заслужено набула популярності та не здає позицій до тепер. Якщо *Java* розрахована на кросплатформність, *C#* на екосистему *Windows* та полегшення синтаксису при розробці, то *C++* за універсальність напрямів розробки програм та за надавання контролю до системних ресурсів.

Розглянувши ці три мови програмування було обрано мову *C++* для створення програмного коду ГІС, так як *C++* наразі є домінуючою в системному програмному забезпеченню, настільних, серверних та мобільних додатках, де є підвищені вимоги до швидкодії та використання системних ресурсів. Це дає більше можливостей для підтримки та розвитку проєкту в майбутньому, а також упереджує від мігрування розробки на інші мови, що може зайняти багато часу, ресурсів, та в цілому погіршити якість системи.

В якості інтегрованого середовища розробки (*integrated development environment - IDE*) вибрано *Microsoft Visual Studio* 2017. Це зумовлено тим, що дане *IDE* має в наборі весь мінімально потрібний інструментарій для роботи над кодом, в тому числі можливість застосувати додаткові надбудови, які розширяють

можливості середовища розробки. Особливо виділити *Visual Studio* 2017 можна за дуже корисну можливість автодоповнення та генерації коду, великої кількості детально описаної офіційної документації та наявність зручного редактору коду.

3.2. Стани геоінформаційної системи

Для розробки ГІС потрібно сформулювати стани системи при взаємодії з користувачем через графічний інтерфейс та при аналізі даних трекінгу.

Перш за все це, має бути стан виведення цифрової мапи. Це основний стан програми, який надає уявлення користувачеві про положення потрібних точок на мапі у вигляді графічних позначень, в тому числі позначення місцезнаходження електромобіля відносно станції, яка надається ГІС як найбільш привілейована для відвідування з поточним запасом ходу авто. Такий стан програми включає й аналіз даних трекінгу й візуалізацію даних про положення конкретної станції.

Для того, щоб зібрати потрібні дані для аналізу, крім застосування вхідних даних з *GPS* про положення автомобіля-споживача, та даних трекінгу станцій, потрібні ще дані про сам автомобіль, так як саме ці дані перевіряються, порівнюючись з характеристиками зарядних станцій. Звісно, такі дані можуть бути надані як тестові набори, на основі яких тестується ГІС перед безпосередньою експлуатацією, але для того, щоб додати більшу гнучкість системі, було прийнято рішення спочатку надавати вибір користувачеві щодо електрокару з початкового набору, який з оновленнями буде розширений. Такий крок дає користувачам дає більше свободи у взаємодії з системою, так як пропускає автоматичну ідентифікацію авто, що на початкових етапах розробки є не пріоритетним завданням, яке в подальшому буде вимагати безпосереднього доступу до ПЗ бортової електроніки. Варто зазначити, що такі випробування мають місце на перспективу розвитку даної ГІС, але для початку будуть використовуватися ручний вибір моделі авто користувача. На основі лише вибраної назви моделі, всі характеристики авто, які потрібні для підбору станції будуть завантажені для аналізу автоматично. Для такого вибору моделі авто слід виокремити стан системи,

який скоріш за все буде початковим, так як напряму зв’язаний з вибором початкових даних для подальшого аналізу.

Третій стан буде виділений для додаткової інформації про станцію чи авто. Це інформаційний екран, на якому зосереджені важливі дані щодо станції та авто. Якщо стан мапи дозволяє в основному побачити лише розташування географічних об’єктів на мапі, що займає весь екран, то для перегляду додаткових інформаційних панелей застосовується саме окремий стан.

Тобто ГІС буде мати такі три стани:

– модель електрокара;

– цифрова мапа;

– додаткова інформація.

Саме на ці стани потрібно орієнтуватися при написанні програмного коду ГІС [12]. Перш за все варто розробити програмний механізм перемикання та зберігання станів ГІС.

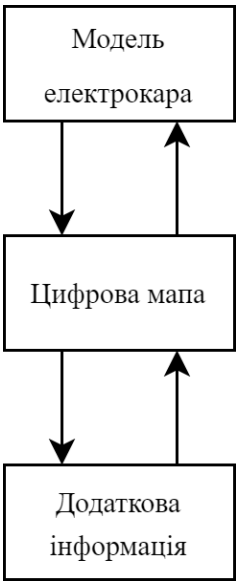

Рис. 3.1. Діаграма переходів між станами ГІС

З програмної точки зору, кожен стан створюється як окремий об’єкт та за вибором вивантажується чи тримається в оперативній пам’яті ПК. Це зроблено за допомогою програмного стеку, де стани за потребою зберігаються. Наприклад, для першого стану (рис. 3.1.) утримання в стеку після переходу не має потреби, так як

він потрібен для задання частини початкових даних, і при роботі з другим та третім станами перший непотрібен взагалі, так як повернення до нього означає нові початкові дані, а отже – нову ітерацію аналізу даних трекінгу. Другий стан потрібно утримувати в стеці при переході в третій, так як другий стан є найголовнішим в програмі та користувач отримує з нього дані про положення на місцевості. Третій стан це лише інформаційна панель для показу тієї інформації, яка б візуально заважала взаємодії користувача зі станом цифрової мапи. Для того, щоб не перевантажувати користувача додатковими даними на одному екрані з мапою, створюється можливість переходу до окремої інформаційної панелі, яка не має ділити увагу зі зображеннями географічних об’єктів на певній місцевості. Отже, нема сенсу розміщувати об’єкт третього стану в стек, так як з ним алгоритмічно не зв’язані окремі розрахунки щодо даних трекінгу, як це зроблено у випадку з другим станом, під час якого й здійснюється аналіз по описаних в другому розділі кроках алгоритмів. Клас об’єктів стану називається *State*, який використовує декілька віртуальних методів для підтримки відтворення станів:

– *start();*

– *control();*

– *update();*

– *show().*

Й два віртуальні методи для керування станами ГІС:

– *freeze();*

– *resume().*

Метод *start()* призначений для того, щоб створити та ініціалізувати стан. В ньому проводиться задання початкових значень аргументів через присвоєння їм вже початково визначених значень, наприклад координати географічного об’єкту за замовчуванням приймають нульові значення для горизонтальної та вертикальної осі. Але є й випадки, коли атрибутам присвоюються значення, які надані іншими станами або зчитуються із зовнішніх джерел, на кшталт текстових файлів. Метод *start()* використовується лише один раз для створення об’єкту класу *State*, в разі

поміщення стану в стек, метод *start()* не застосовується для поновлення роботи стану, так як стан вже створено, збережено, але ще не видалено.

При кожному створені стану класу *State* застосовується й створення об'єктів, які завдячують класам мультимедійній бібліотеці *SFML*. Хоча вона більше орієнтована на графічну взаємодію з користувачем, не слід забувати, що саме ця бібліотека відповідає за створення віконного інтерфейсу з програмою, тобто впроваджувати використання коду, який використовує класи *SFML* потрібно ще на етапі розробки станів ГІС, тобто ще навіть до фактичного створення графічного інтерфейсу. Враховуючи це, слід використати на етапі ініціалізації станів такий клас, як *sf::Clock*, що надає поняття «тіків» при виконанні програми, тобто проміжків часу, коли стаються програмні розрахунки. Це потрібно не лише для таймерів чи секундомірів, а для взаємодії з програмою в реальному часі загалом, для чого вимір проміжків часу передбачений не лише в секундах, а й в мілі- й навіть в мікросекундах, зважаючи на різні потреби при затримці ручного введення даних.

Крім початку відліку часу для кожного стану при ініціалізації також дуже часто будуть створюватися об'єкти текстового виводу за класом *sf::Text*. Даний клас відповідає за роботу з текстом не лише як класичні типи *string*, які відповідають за саме збереження символьних масивів, а й з різними додатковими методами й атрибутами.

Наприклад, щоб задати вміст рядка, використовується метод *setString()*, куди передається якраз потрібний для зберігання символьний масив. Для того, щоб обрати розмір виведення символів, використовується метод *setCharacterSize(),* аргументом для якого є цілочисельне значення в пікселях. Для вибору шрифту, який представлятиме раніше заданий текст через *setString()* використовується метод *setFont()*. Щоб задати колір тексту, застосовується метод *setFillColor()* який, в свою чергу, приймає як аргумент об'єкт класу *sf::Color*.

В класі *sf::Color* через власний колір може задаватися як з переліку деяких широковикористовуваних кольорів (*sf::Color::Black, sf::Color:Red, sf::Color::White*, тощо), так і на основі параметрів значень колірної моделі *RGB* (*red, green, blue* — червоний, зелений, синій), яка заключається у відтворенні кольорів

на основі різної інтенсивності показу трьох перерахованих. Це перші три аргументи для конструктора класу *sf::Color*, четвертим параметром є значення альфаканалу, або простими словами – прозорістю колірної зони. З використанням параметрів надається більша гнучкість налаштування відображення кольору, в поєднанні з методом *setFillColor()* в зазначеному прикладі – регулювання кольору й прозорості тексту.

Також, в методі *start()* крім ініціалізації тексту відбувається й ініціалізація текстур. Текстура (клас *sf::Texture*) – це зображення, яке використовується надалі для створення об'єктів у програмі ГІС, які мають власні координати та властивості. Такі об'єкти називаються спрайтами. Спрайти можна змінювати по двовимірним координатам *x* та *y* за розміром, але в такому разі якість зображення буде пікселізуватися, тобто гірше виглядати через брак візуальної деталізації.

Розглядаючи наступний метод відтворення станів *control()*, то він відповідає саме за введення даних від користувача у вигляді взаємодії з віртуальними (екранними) чи фізичними кнопками пристрою введення. *SFML* надає власний клас *sf::Event* який створений для прослуховування наявності натискання фізичних кнопок пристрою введення, а *sf::Keyboard* відповідає за визначення коду певної кнопки, наприклад *sf::Keyboard::Enter* надасть код кнопки *Enter* на клавіатурі ПК. В поєднанні *sf::Event* з *sf::Keyboard* можна визначити, чи була нажата кнопка із заданим кодом.

Метод *update()* потрібен для оновлення програмної логіки. Для цього під час виконання здійснюються циклічні перезапуски внутрішнього програмного таймера, на відміну від зовнішніх потреб, наприклад, надання статистики про час виконання програми, коли секундомір зупиняється лише при виході з ГІС, внутрішній програмний таймер є основним мірилом часу для виконання всієї програми, але при тому, зважаючи на те, що в ГІС є декілька станів, які являють собою окремі об'єкти з окремим часом існування, внутрішній програмний таймер запускається для кожного стану окремо, але неодночасно з іншими, так як тоді без підтримки мультипоточності виконання це було марною тратою апаратних ресурсів. Крім того, створення відліку часу на основі таймера, тобто зворотного

відліку часу краще логічно розуміється і сприяє додатковим смисловим захистом від переповнення декрементного лічильника. Крім цього, в даному методі можна розміщувати інші події, які потребують мінімальної затримки відповіді системи, до прикладу – пересування над цифровою мапою, де затримки при введенні найменш допустимі.

Для того, щоб покадрово відтворювати графічне зображення ГІС застосовується метод *show().* Саме цей метод відповідає за відтворення візуального стану модуля на екрані. Поскільки бібліотека *SFML* надає програмні інструменти для відтворення зображення у графічних вікнах через власні класи, логічно було б відокремити такі засоби у окремий програмний метод, який буде використовуватися для кожного стану програми. Щодо зазначеного покадрового відтворення, то через внутрішній програмний таймер за допомогою *sf::Clock* здійснюється покадровий вивід зображення з частотою 30 кадрів за секунду, або ж 30 *FPS* (*frames per second* - кадрів за секунду). Така частота забезпечує плавність відтворення руху при зміні кадрів. За створення кадрів на екрані відповідає метод *display()*, а метод *draw()* – за переміщення підготовлених графічних об’єктів на кадр для відтворення. Щоб кадри оновлювалися, з вказаною частотою, що відміряється через внутрішній програмний таймер застосовується метод *clear()*, який очищує екран від попереднього кадру. Наприклад, застосовуючи клас *sf::Color*, можна очистити екран від кадру, заповнивши при цьому пустий візуальний простір певним кольором, як-от *window.clear(sf::Color::Black)* змінює кадр зображення на повністю чорний колір екрану. Це потрібно, щоб при наявності прозорості на раніше показаних кадрах наступні не накладалися на попередній, виключаючи непередбачений вигляд зображення для користувача.

Метод *freeze()* використовується для зупинки роботи стану зі збереженням його значень без видалення.

Метод *resume()* створений для поновлення роботи раніше збереженого стану при роботі з ГІС.

3.3. Аналіз даних в геоінформаційної системі

Аналіз даних трекінгу починається після створення та ініціалізації другого стану ГІС, тобто цифрової мапи. Після того, як були введені дані про модель автомобіля через натискання відповідної графічної кнопки з назвою моделі, дані про автомобіль підвантажуються з простої БД текстового файлу *CarInfo.txt* в структуру під назвою *InfoBuf*. В своєму складі ця структура має такі дані про авто, як фото, тип роз'єму для зарядки, та іншу інформацію, яку надає виробник. Така БД має впорядковані за розташуванням в текстовому форматі дані про авто, які підтримує ГІС. Крім того, підвантажаються деякі статистичні дані, надані або самим користувачем, або зовнішніми джерелами щодо використання певної моделі авто. До таких належать середній час повного розряджання елементу живлення під час руху, середня швидкість авто, середній час зарядки. Поскільки ці дані належать до електромобіля, який є одним з двох основних географічних об'єктів (*Gobjects*) в ГІС, то така інформація належить до типу «Інформаційний опис» географічного об'єкту «Споживач» з атрибутами *photo, connector, av_chrg_time, av_speed, av_dischrg_time*. Зчитування в структуру *InfoBuf*, а не напряму ініціалізуючи значення з текстового файлу дає змогу мати проміжну ланку для перевірки інформації, та отрмання даних з декількох джерел, наприклад, зчитування з декількох текстових файлів, які представляють характеристики від виробників та від користувачів у вигляді середніх значень. Схоже відбувається й зчитування даних щодо даних трекінгу джерел поповнення енергії, але використовується структура з полями, які потрібні для вибору зарядних станцій, наприклад доступність, що означає чи станція взагалі в робочому стані – змінна типу *boolean,* популярність серед користувачів – цілочисельне значення в діапазоні від 1 до 5, ідентифікаціний номер станції, бренд, наявні типи конекторів для зарядки, кількість місць для зарядки й таке інше. Зарядні станції є також географічними об'єктами *Gobject*, де вищеперелічені дані також можуть підвантажуватися з зовнішніх джерел, наприклад, текстового файлу в структуру *InfoBuf*, а після перевірки – до відповідних атрибутів географічного об'єкту. Згідно з моделлю

ExIFO2, інфомаційний опис належить до непросторового абстрактного типу, тобто *Aspatials*, до якого входить чіткі та нечіткі представлення даних. До чітких можна віднести текстову інформацію, як ідентифікаційні послідовності чи назви моделей авто, або числові значення, які представляють чітко визначені величини, або які умовно так приймаються, наприклад час повного розряду акумулятора, швидкість електромобіля.

Крім типу «Інформаційний опис», також застосовується й тип «Геодані», від якого походить *Geo*, що містить в собі інформацію про місцезнаходження різних географічних об’єктів.

Координати Gobject мають як довідковий, обчислювальний, так візуальний характери. За географічні координати відповідає змінна *geo_latitude*, *geo_longitude,* які представляють довідковий характер й не використовуються при обчисленнях на мапі. Змінні *geo_X* та *geo_Y* відповідають за зберігання координат точок в метрах та використовуються для обчислень відстаней між точками та їх розташуванням. Змінні *onScreen_X* та *onScreen_Y* призначені для зберігання координат графічної частини *Gobject*, тобто зображення, яке пов'язане розташуванням з іншими елементами графічного інтерфейсу.

Крім представлення координат точок, також має бути атрибут для збереження унікальності точки – це виконує атрибут *ID*, що дає порядковий номер географічному об’єкту. Якщо для означення електромобіля це не так важливо, то для переліку станцій для порівняння це потрібно. Інші типи геометричних примітивів типу «Лінія» чи «Область» поки не застосовуються, але концептуальна модель підтримає їх введення у програму в перспективі оновлень.

Для визначення станції, яка найбільше підходить для авто з певним процентом заряду можна було б проаналізувати відразу всі станції, які наявні на цифровій мапі ГІС. Але мапа може охоплювати різну територію, де може бути велика кількість зарядних станцій для електрокарів, тому було вирішено для меншого використання ресурсів системи обирати лише географічні об’єкти станцій, що лежать в зоні досяжності електромобіля з поточним зарядом акумулятора. Зона досяжності являє собою круглу область в двовимірному

просторі на мапі, яка має радіус, рівний довжині шляху автомобіля в ситуації коли потрібно проїхати по прямій на середній швидкості до повного розряду акумулятора й, як наслідок, припинення руху авто. Для ГІС, яка має в розпорядженні лише координати точок на карті та має ідентифікатори квадратних фрагментів мапи, які створюють собою сітку та полегшують знаходження об'єктів, так як кожен квадрат – це об'єкт класу *square_zn*, який має в переліку власних атрибутів не лише ідентифікатор та дані про власні координати, а й вектор рядків *Stations,* який складається з переліку ідентифікаторів станцій, які лежать в даному квадраті (вектор – це структура даних в мові програмування, яка являє собою динамічний масив, що може керувати виділеною для нього пам'яттю). Таке опитування допомагає не перевіряти мапу поточково, а лише опитувати окремі квадрати на те, чи є вектор *Stations* пустим через метод *isSquareEmpty()*.

Для того, щоб виявити зону досяжності авто, застосовується алгоритм з рис. 3.2. В алгоритмі формується область пошуку в межах так званої «зони досяжності» авто, яка представляє собою круглу зону з центром в точці положення авто на мапі, а радіусом рівним довжині прямолінійного шляху L (довжина максимального шляху) даного авто з середньою швидкістю $U_{сер}$ та часом розряду акумулятора з поточного відсотку до нуля t_2. Радіус $R_{досяжн.}$ є рівним довжині прямолінійного шляху $L_{макс.шляху}$ цього авто з середньою швидкістю $U_{сер}$ та часом розряду акумулятора з поточного відсотку до нуля t_2.

Розрахунок радіусу зони досяжності електромобіля обраховується за формулами (2.1 - 2.3).

$$t_2 = T_{розр.} - t_1; \tag{2.1}$$

де $T_{розр.}$ – час повного розряду елемента живлення конкретного електромобіля при усередненому використанні, t_1 - час розряду елемента живлення конкретного електромобіля при усередненому використанні від 100% до поточного відсотку заряду акумулятора.

$$L_{макс.шляху} = U_{сер} \cdot t_2; \tag{2.2}$$

Рис. 3.2. Алгоритм визначення зони досяжності авто

де $U_{сер}$ – значення середньої швидкості електромобіля, t_2 – час повного розряду акумулятора з поточного відсотку заряду.

$$R_{досяжн.} = L_{макс.шляху}; \quad (2.3)$$

де $R_{досяжн}$ – радіус зони досяжності, $L_{макс.шляху}$ - довжина прямолінійного шляху авто до повного розряду елементу живлення без підзарядок з поточного проценту заряду.

Коли L знайдено, шукається початковий квадрат, що умовно позначається «1». Початковий квадрат – це той квадрат на сітці мапи, де знаходиться

електромобіль на момент початку аналізу даних. Коли координати зіставлені, та такий квадрат знайдено, починається розширення зони, яку створюють оточуючі квадрати (рис. 3.3.).

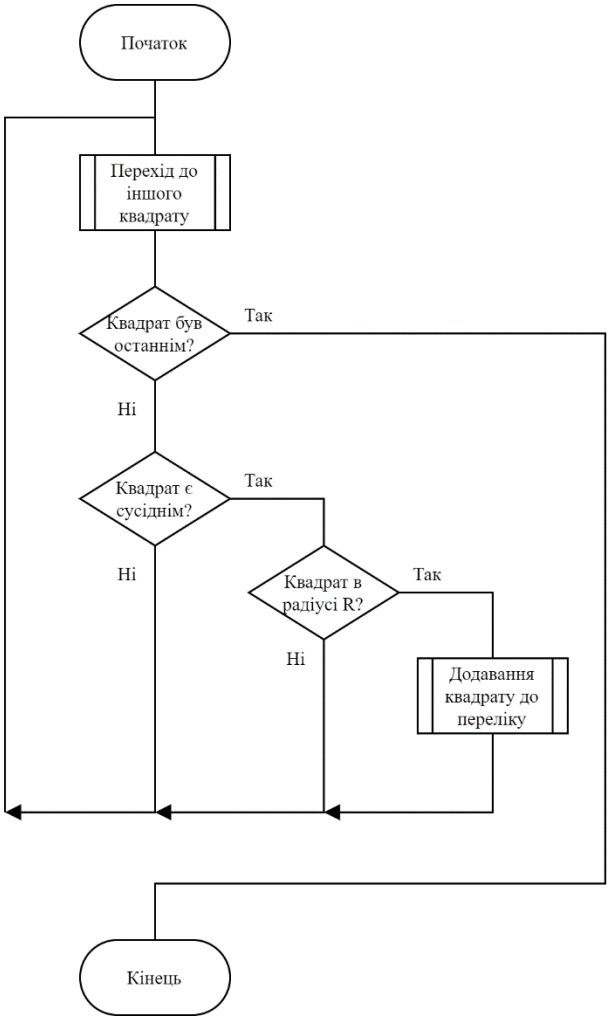

Рис. 3.3. Алгоритм пошуку оточуючих квадратів для зони досяжності авто

Таким чином, квадрати додаються шаром за шаром до тих пір, поки жоден квадрат з нового оточуючого шару не буде лежати в радіусі зони досяжності $R_{\text{досяжн}}$, яка в свою чергу рівна значенню L. Якщо, наприклад, маємо цифрову мапу величиною в 12 × 7 квадратів. На такій мапі є географічні об’єкти – авто (точка O) та 6 станцій зарядки (точки A, B, C, D, E), які потрібно порівняти. По представленим вище алгоритмам (рис. 3.3., рис. 3.3.) за радіусом зони досяжності авто формується зона через найближчі квадрати. На першому етапі визначається початковий квадрат «1», де знаходиться авто. Потім такий квадрат поступово оточується сусідніми, поки хоч входить до радіусу досяжності $R_{\text{досяжн}}$. Поступово додавання шарів до

зони, тобто її розширення, видно з позначень таких шарів у хронологічному порядку – квадратів з позначками «2» й «3» відповідно. Відповідно, коли ріст квадратної зони припиняється, інші квадрати, так як і станції, виключаються з подальшого аналізу.

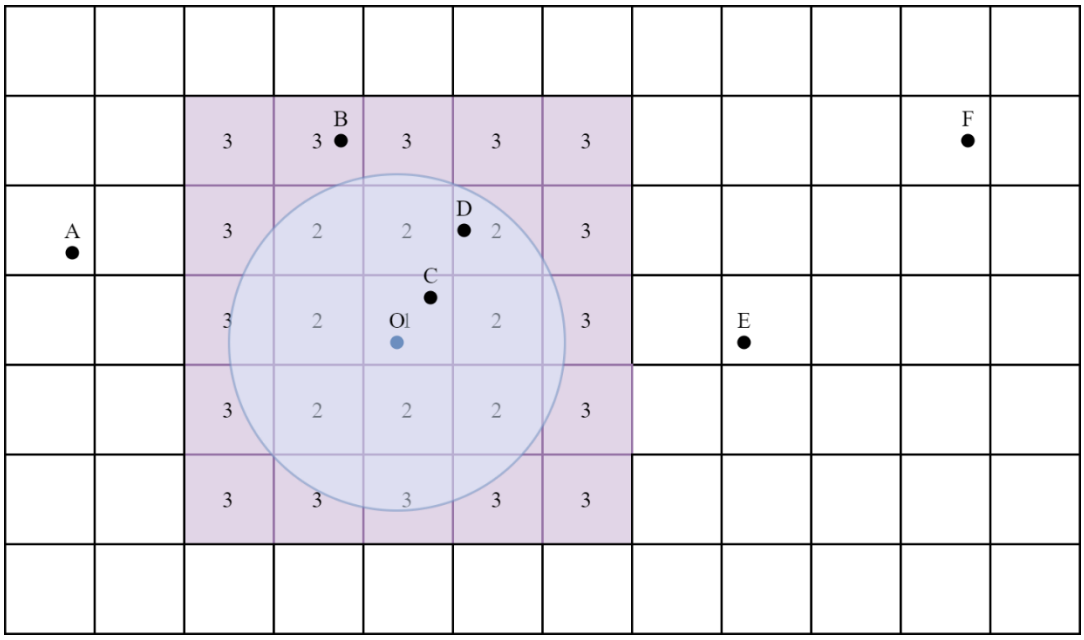

Рис. 3.4. Схема першого етапу знаходження станцій в зоні досяжності авто

На рис. 3.4. прозорим блакитним колом позначено уявний круг зони досяжності через круглу форму, з якою програма взаємодіє виключно через координати точки *O* та довжину $R_{досяжн}$. Фіолетовим виділена та зона досяжності з квадратів, на якій зупинилися ітерації розширення зони.

На другому етапі через метод *isStationInR()* перевіряється відстань між точкою *O* до інших точок, що залишилися, тобто має виконуватися твердження $OX <= L$, де *O* – точка місцезнаходження авто на мапі, *X* – точка місцезнаходження станції в квадратній зоні досяжності на першому етапі, *L* – довжина максимального шляху, що рівна $R_{досяжн}$. На рис. 3.5. зображені зеленим ті квадрати, для точок станцій яких метод *isStationInR()* повернув значення *true*. Червоний квадрат той, який хоч і увійшов до зони досяжності на першому етапі, на другому був відкинутий для подальшого аналізу.

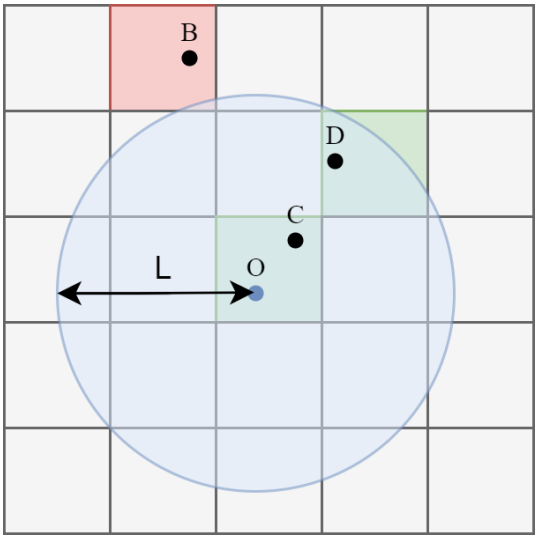

Рис. 3.5. Схема другого етапу знаходження станцій в зоні досяжності авто

В результаті з точок *A, B, C, D, E* залишилося лише дві точки *C* та *D*, які лежать в зоні досяжності авто при ідеальних умовах для пересування. В умовах великої кількості станцій на мапі такий відбір станцій може суттєво скоротити кількість кандидатів для подальшого аналізу.

Далі відбувається перевірка типу конекторів на поточних станціях-кандидатах. Для цього співставляються дані про конектори зі структур *InfoBuf* географічних об’єктів авто та станцій. Враховуючи, що станції можуть мати декілька терміналів з різними роз’ємами, перевірка однієї станції може проходити декілька ітерацій. За порівняння відповідає метод *checkConnectors()*, що порівнює конектори з відповідних атрибутів та повертає вектор рядків з тими роз’ємами, які підходять для зарядки поточного авто. Також, в методі присутня поглиблена перевірка при наявності роз’єму типу *CCS Combo* в електромобіля, що дозволяє при наявності терміналів для *Type 1* чи *Type 2* зарядити авто через інший, старіший тип з’єднання звичайним режимом зарядки. Також, враховуються наявні режими зарядки, і віддається перевага вищому значенню номера режиму, тобто швидкості зарядки, якщо таке тип роз’єму електромобіля. Загальний алгоритм роботи відбору станцій для авто за типом роз’єму зображено на рис. 3.6.

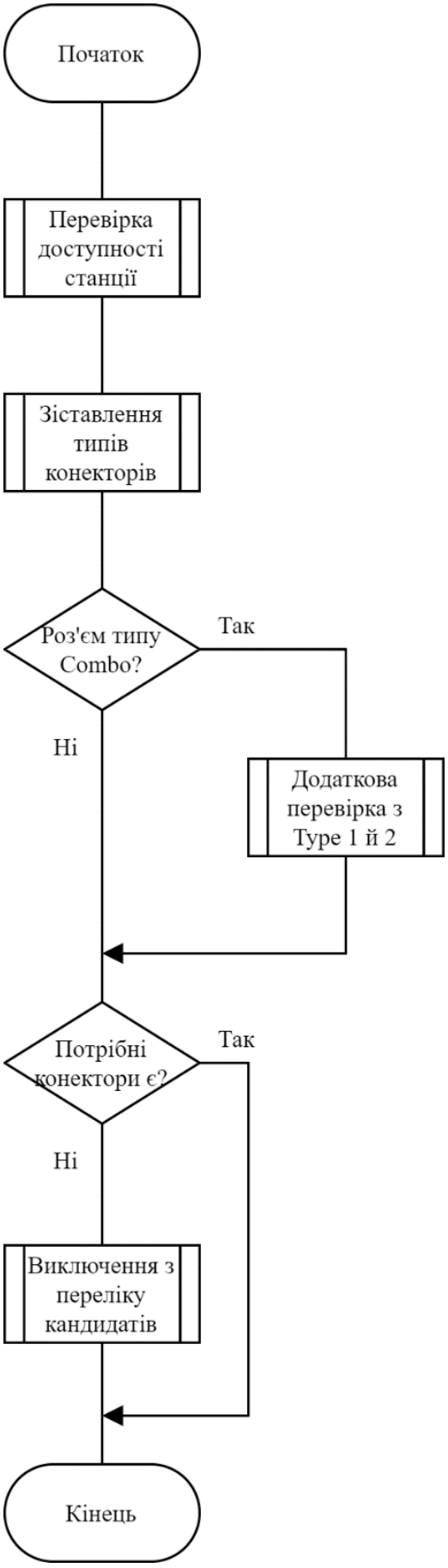

Рис. 3.6. Загальний алгоритм роботи відбору станцій для авто за типом роз’єму

Після такого відбору залишаються станції, які можуть обслуговувати конкретне авто в даний момент часу. Тобто можна було б доїхати до одної з декількох станцій в тому випадку, якщо після відбору через зону досяжності та типом роз'єму залишається більше, ніж одна станція у списку кандидатів, яку зберігає вектор *Candidates_st_list*. Таким чином, якщо станцій у списку не залишилося, про це повідомляється у відповідному повідомленні для користувача через метод *warn_message()*. Якщо кандидат лише один, додаткових порівнянь не проводиться і користувачеві виводиться область цифрової мапи, на якій видно як доїхати до станції зарядки із візуалізацією маршруту, який показується на основі запиту до *GPS*-сервісів на основі вузлових точок. Якщо кандидатів декілька, проводиться додаткова перевірка.

Додаткова перевірка полягає в тому, що з тих станцій, що залишилися після попередніх перевірок порівнюють такі їх дані, як довжина маршруту та режим зарядки. Перевірка створена в методі *add_check()* приймає оновлений вектор станцій-кандидатів та повертає ідентифікатор тієї станції, яка вважається найкращим вибором за одним чи двома додатковими критеріями. Ці два критерії зберігають змінні *route_len* та *charge_md.* Перша змінна зберігає опитане з GPS-сервісу значення довжини маршруту, що є пріоритетним критерієм, тобто якщо знаходиться одне мінімальне значення серед кандидатів, перевірка режиму зарядки не проводиться. В разі, якщо знаходяться декілька мінімальних значень довжини маршруту, які рівні між собою, проводиться друга перевірка по змінним *charge_md,* які беруться зі структури *InfoBuf*. В даному разі, чим більше значення режиму (від 1 до 4), тобто чим швидша зарядка, тому кандидату й надається перевага у виборі як рекомендованій за аналізом, здійсненим ГІС, станції. Якщо ж навіть після цього залишаються станції з такими однаковими даними трекінгу, тоді обирається та, яка розташована перша в списку *Candidates_st_list*.

Тоді ідентифікатор обраної станції передається до методу об'єкту класа *Map,* що називається *showPathArea()*, який візуалізує ту область мапи, в якій знаходяться авто та обрана після аналізу станція, разом з іншими позначками на мапі, потрібними для користування другим програмним станом ГІС.

Для розрахунку запасу ходу авто використовується формула (2.4).

$$L_{\text{зап.ходу}} = \frac{W_{\text{пот.}}}{W_{100}} \cdot 100 \ ; \qquad (2.4)$$

де $L_{\text{зап.ходу}}$ – запас ходу електромобіля в кілометрах, $W_{\text{пот.}}$- енергія акумулятора в певний момент часу, що характеризує його ємність (кВт * год), W_{100} – енергія акумулятора на 100 км ходу (кВт * год).

Щоб виміряти процент заряду акумулятора, застосовано формулу (2.5).

$$x_{\text{зал.}} = \frac{W_{\text{пот.}}}{W_{\text{макс.}}} \cdot 100\% \ ; \qquad (2.5)$$

де $x_{\text{зал.}}$ – поточне значення заряду у відсотках, $W_{\text{пот.}}$- енергія акумулятора в поточний момент часу, що характеризує його ємність (кВт * год), $W_{\text{макс.}}$ – енергія повністю зарядженого акумулятора, що характеризує його максимальну ємність (кВт * год).

Середня швидкість для легкового електромобіля за замовчуванням знаходиться по значеннях обмежень швидкості руху на дорогах для легкових автомобілів, що становлять 50 км/год у населених пунктах, 20 км/год у житлових і пішохідних зонах, 90 км/год поза населеними пунктами, 110 км/год на автомобільній дорозі з окремими проїзними частинами, що відокремлені одна від одної розділювальною смугою, 130 км/год на автомагістралі. Знайшовши середнє арифметичне цих значень встановлюється середня швидкість руху легкового електромобіля.

$$\upsilon_{\text{сер.}} = \frac{50 + 20 + 90 + 110 + 130}{5} = 80 \text{ (км/год)} \ .$$

Час розряду акумулятора, якщо не надано статистичних даних щодо часу активного використання авто, визначається за формулою (2.6).

$$t_{\text{розр.}} = t_2 = \frac{L_{\text{зап.ходу}}}{\upsilon_{\text{сер.}}} \ . \qquad (2.6)$$

Таким чином вираховуються значення орієнтовного часу розряду батареї авто та його поточного відсотку заряду для показу користувачеві в інформаційних панелях ГІС.

Аналіз підбору станції для зарядки авто по даним трекінгу джерел поповнення енергії полягає в:

1) Знаходженні теоретичної зони досяжності авто, радіусом якої є запас ходу авто в кілометрах, і яка використовується для формування зони з територіальних квадратів.

2) Ітеративному розширенні зони з територіальних квадратів на мапі й внесенні до списку належних до них станцій.

3) Порівнянні радіусу зони досяжності авто зі значенням переміщення від авто до станції.

4) Перевірці станцій на доступність, тобто чи взагалі станція є працездатною.

5) Перевірці станцій на сумісність з роз'ємом моделі авто-споживача.

6) Порівнянні довжини шляху від споживача до джерела поповнення енергії.

7) Перевірці на наявність у станцій з сумісним роз'ємом швидкої зарядки, зазвичай це супроводжується прямою сумісністю сучасних зразків роз'ємів живлення.

В результаті аналізу даних трекінгу ГІС надає користувачеві інформацію про обрану на основі відбору зарядну станцію з подальшою візуалізацією на цифровій мапі.

3.4. Візуалізація даних в геоінформаційній системі

Спершу відбувається завантаження ресурсів для візуалізації. Основними такими ресурсами є зображення й шрифт, це – файли форматів *png* та *jpeg*, а також ресурси для тексту – це шрифти, що мають формат файлів *otf*.

Формат зображень *png* (*portable network graphics*) та *jpeg* (*joint photographic experts group*) не вдаваючись до технічних подробиць відрізняються тим, що перший має більші розміри файлу при збереженні та використовує стиснення без втрат якості зображення, а другий формат має хоч і менший розмір зберігання, але використовує стиснення з втратами якості зображення. Крім того, конкретно для розробки візуалізації ГІС, *png*-формат зображення підтримує збереження

прозорості, що дуже потрібно при роботі з зображеннями, які мають візуально геометрично відрізнятися від прямокутника чи квадрата.

Формат шрифту *otf (OpenType Fonts)* потрібен для відображення текстової інформації у різних графічних оформленнях. Такий формат був вибраний через більші можливості для зберігання символів (до 65 000), що дає більше свободи при оформленні текстової інформації та доповненні шрифтів.

Хоча *SFML* й передбачає завантаження зображень та шрифтів через метод *loadFromFile(),* але зазвичай потрібно працювати з різними форматами, а на етапі ініціалізації може настільки багато таких завантажень, що краще створити клас, який, по-перше – дасть розробнику можливість розрізняти завантаження зображень та шрифтів, а по-друге – зробить операції завантаження в пам’ять цих даних та присвоєння їх відповідним об’єктам *SFML* через різні методи, в тому числі для того, щоб на етапі присвоєння не доводилося додатково перевіряти формати файлів. Для цього створено користувацький клас *AssetManager*, що має окремі методи для завантаження даних в пам’ять – *loadTexture()* для зображень та *loadFont()* для шрифтів. Аргументами даних методів є назва ресурсу, що використовуватиметься для доступу до нього в програмі, а другий аргумент це фактичний шлях у файловій системі до ресурсу для завантаження, обидва аргумента рядкового типу. Також *AssetManager* має два методи для повернення ресурсів у розпорядження ГІС для візуалізації, це методи *getTexture()* та *getFont()*.

Сформувавши допоміжний клас для полегшення роботи з ресурсами програми, варто звернути увагу на розробку елементу інтерфейсу, через які користувач зможе взаємодіяти з ГІС. Такими елементами є графічні кнопки, що створюються з перспективою не лише взаємодії з користувачем через такі пристрої введення, як комп’ютерна миша, а й через сенсорний екран, який зазвичай має лише наекранні елементи інтерфейсу для вводу. Графічна кнопка має бути окремо оформленою областю екрану у вигляді інтерактивного об’єкту, який візуально реагує на положення курсору в зоні знаходження такої кнопки, а також при натисканні лівої фізичної кнопки миші в зазначеній зоні на екрані. Кнопка може бути оформлена як просте зображення дії чи символа, або як об’єкт зі складнішою

структурою у вигляді фонового зображення графічної кнопки, яке зазвичай геометрично зівпадає з інтерактивною зоною для курсора. Для складнішої структури графічної кнопки також можливе застосування окремого параметра, який дозволяє налаштовувати символьний вміст в межах зони кнопки.

Всі звичайні графічні кнопки працюють в проєктній ГІС за алгоритмом на рис. 3.7. Графічні кнопки також є дуже важливим складовими графічного інтерфейсу не лише в індивідуальному значенні, а й в плані груп. Це зумовлено тим, що поскільки графічні кнопки мають активувати програмний тригер для визначеної для натискання подальшої послідовності подій. Наприклад, після натискання на кнопку «Додому» програма переходить до стану головного меню, або при натисканні на кнопку «Вимкнення» здійснюється вихід з програми.

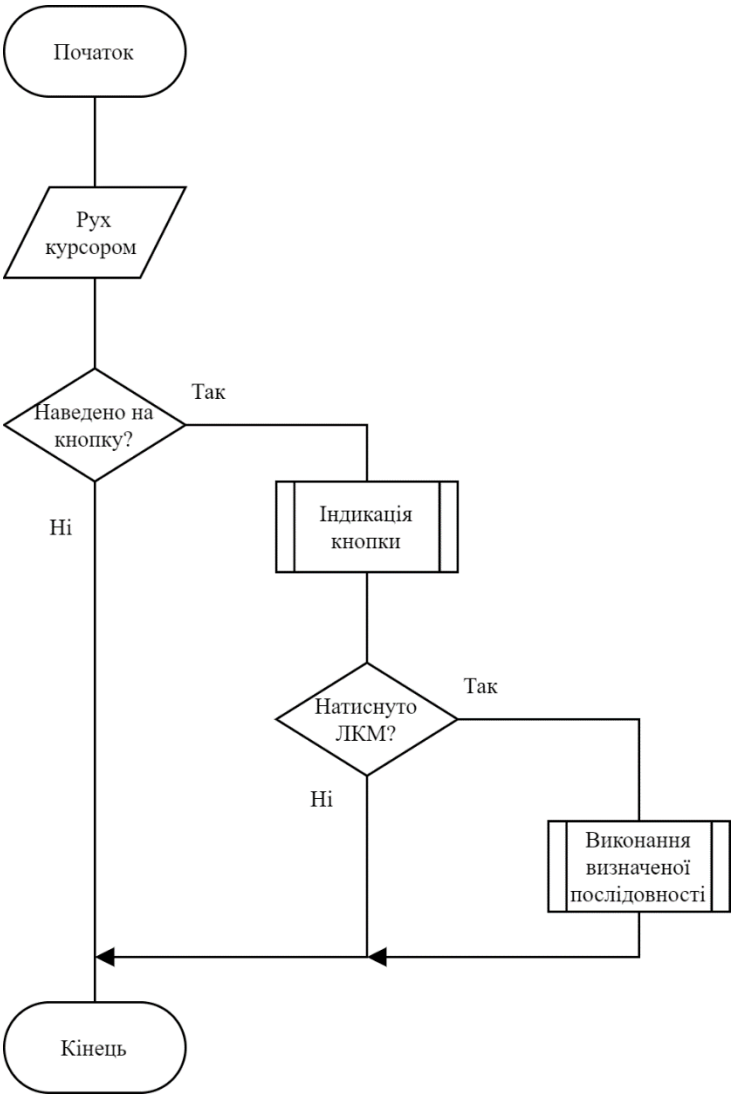

Рис. 3.7. Схема алгоритму реагування кнопок на положення курсору

Але часто бувають випадки, коли користувачу потрібно надати декілька варіантів подібних дій, наприклад налаштування програми. До цього відноситься гучність звукових ефектів, яскравість зображення, зміна масштабу інтерфейсу, можливо навіть стилю чи розташування кнопок, але також графічні кнопки можуть відповідати за зміну шаблонів поведінки прогами в цілому, наприклад – якщо користувачеві не потрібно використовувати певний режим, або навпаки – потрібно його увімкнути, то графічні кнопки слугують елементом взаємодії і для цієї мети. І якщо кнопки відповідають, нехай, за декілька режимів, або за налаштування певних споріднених параметрів, то тоді є сенс розмістити їх не просто на одному екрані, а ще й згрупувати та розмістити разом.

Програмні меню, як і кнопки, бувають різного складу й вигляду, але всі меню виконують такі задачі, як впорядкування інтерактивних елементів за спрямованістю дій та візуальне полегшення взаємодії з графічним інтерфейсом.

Крім взаємодії з графічними кнопками, треба пам’ятати, що самі кнопки зазвичай являють собою позначення інтерактивної зона на екрані через фонове зображення та текстовий надпис, що показаний за допомогою вибраного шрифту. Ці два елементи опрацьовує клас *AssetManager*, що відповідає за завантаження графічних ресурсів програми в пам’ять і працює за алгоритмом на рис. 3.8. Якщо робота з графічними кнопками вже прописана в коді, це ще не означає, що такі елементи інтерфейсу вже можна застосувати без попередньо сформованого макету розташування елементів інтерфейсу (скорочено MPEI). Створення такого макету є проміжною ланкою в процесі розробки візуальної частини ГІС, так як недостатньо просто назначити інтерактивні елементи, потрібно ще візуально їх згрупувати, в чому й допомагає MPEI. Орієнтуватися при створенні таких макетів потрібно на можливості самої програми та тієї інформації, яка має бути надана користувачеві при відображенні на екрані.

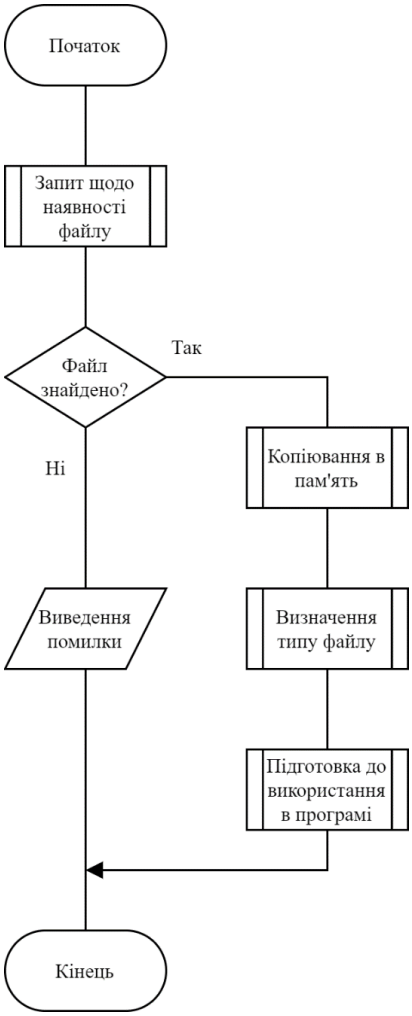

Рис. 3.8. Алгоритм роботи з графічними ресурсами класом *AssetManager*

Також варто орієнтуватися не лише на інтерактивні області екрану, а й на вигляд інтерфейсу загалом, так як при роботі з ГІС важливі не просто графічні кнопки та їх розташування, а й загальна візуальна інформація зі всього інтерфейсу, тобто інформаційні панелі, вигляди різних меню та цифрової мапи в конкретному випадку розробки геоінформаційної системи.

Отже, можливості програми в стислому вигляді презентують раніше описані стани ГІС, тобто:

– модель електрокара;

– цифрова мапа;

– додаткова інформація.

Ці стани допоможуть розробити макети та зрозуміти, які елементи інтерфейсу потрібні в той чи інший момент користування ГІС. Так як станів лише

три, і кожен з них принципово відрізняється за призначенням, варто придивитися до розробки відповідних трьох макетів відображення.

Перший макет стосується стану щодо вибору моделі електрокару. Користувач має за допомогою графічних кнопок обрати з переліку підтримуваних моделей електрокарів обрати ту, для якої потрібно знайти станцію підзарядки. Крім кнопок з назвою моделі авто, слід подумати про виведення такої інформації, як фото вигляду вибраної моделі, її характеристики та, можливо, зображення роз'єму для підзарядки в схематичному спрощеному вигляді. Можливі варіанти розташування наведені на рис. 3.9. - 3.11.

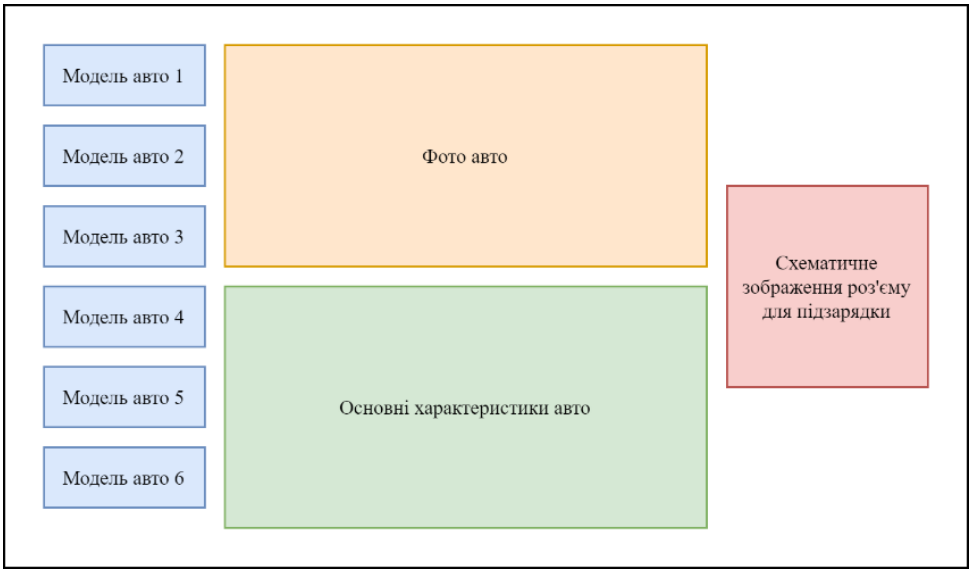

Рис. 3.9. Перший варіант MPEI для стану «Модель електрокара»

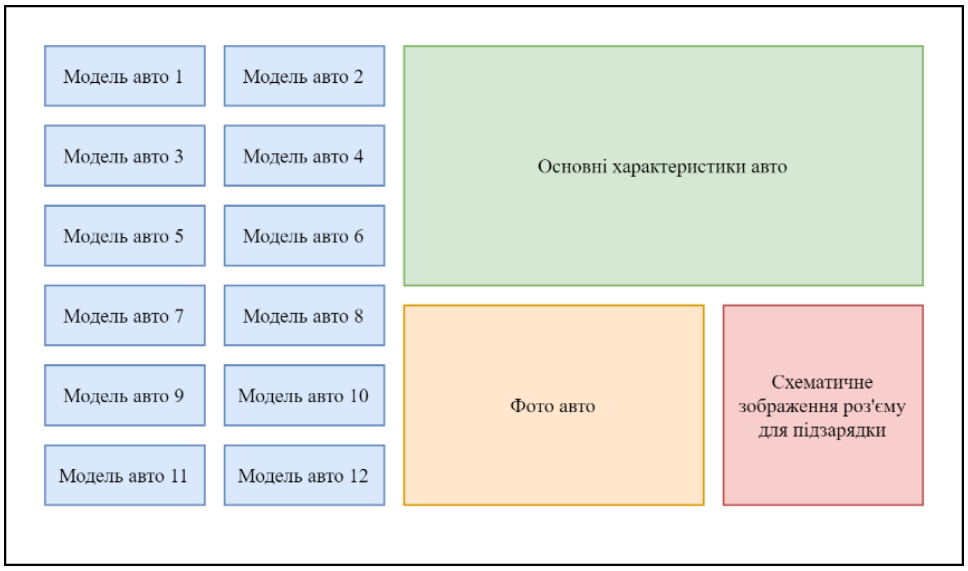

Рис. 3.10. Другий варіант MPEI для стану «Модель електрокара»

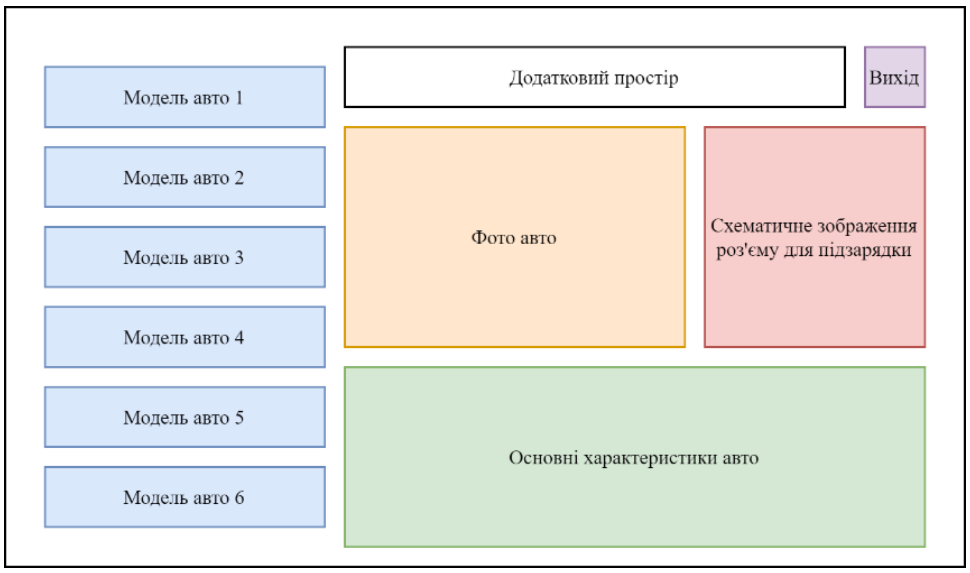

Рис. 3.11. Третій варіант MPEI для стану «Модель електрокара»

Порівнюючи три варіанти візуального представлення першого стану ГІС, перший варіант MPEI є візуально помітно не симетричним та виділяє забагато місця для фото авто, яке є не наскільки важливим по вмісту щоб займати велику область екрану. Графічним кнопкам зліва, які надають користувачеві перелік для обирання потрібної моделі авто, виділяється замало місця на екрані. Через це перший варіант MPEI відкидається й розглядається наступний.

Другий варіант візуального представлення вже симетрично розташовує елементи графічного інтерфейсу на екрані та виділяє більше місця для графічних кнопок, що дозволяє згрупувати їх в дві колонки. Але таке розташування не залишає вільного екранного місця для додавання нових елементів, що буде приносити незручності при додаванні нових можливостей при оновленнях програми і зв’язаних з цим доповненням інтерфейсу. Крім того, велика кількість інтерактивних кнопок впливає на їх простір для текстового відображення назви моделі авто, а отже це може призвести, що шрифт тексту кнопки доведеться зменшувати за розміром, що не бажано для комфортного сприйняття тексту користувачем. Тому цей варіант також потрібно доопрацювати.

Третій варіант враховує недоліки попередніх та пропонує компромісні рішення для розташування елементів графічного інтерфейсу. Також, додана графічна кнопка «Вихід», що спростить вихід з програми, якщо користувач

вирішить завчасно здійснити таку дію. Розміри графічних кнопок були збільшені для можливості зберігання більшої кількості символів, але за рахунок цього кнопок для обрання моделі авто стало знову менше, в чому й полягає компромісність рішення. Так як початково ГІС буде працювати в тестовому режимі, на вхід для аналізу буде надаватися обмежена кількість даних, тому в такому разі розрахунок на невелику кількість підтримуваних для аналізу моделей себе виправдовує, з врахуванням подальших оновлень, і додавання в перелік нових моделей електрокарів та їх характеристик й інших даний, як, наприклад, фото зовнішнього вигляду обраної моделі. Блок, що називається «Додатковий простір» означає той екранний простір, який відведено на перспективу нових графічних елементів, що може містити як інтерактивні елементи, як графічні кнопки, так і неінтерактивні, як, наприклад, коротку інформацію про програму.

За наповненням близько до MPEI стану «модель електрокара» знаходиться третій стан ГІС під назвою «додаткова інформація», що надає дані про обрану станцію зарядки та стан обраного авто. Але оскільки такий стан має надавати саме текстову інформація з деякими фото, то кількість інтерактивних елементів, на відміну від першого стану ГІС має бути меншою, щоб надати якнайбільше екранного простору для інформаційних панелей (рис. 3.12.).

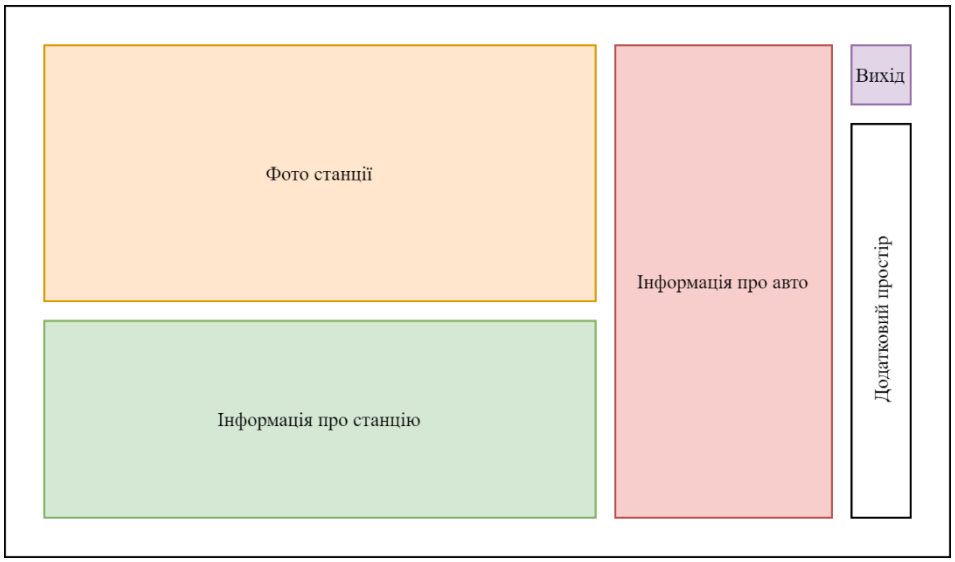

Рис. 3.12. MPEI для стану «Додаткова інформація»

Як видно на рис. 3.12., основні три блоки інтерфейсу: «Інформація про авто», «Фото станції», та «Інформація про станцію» займають саме інформаційні панелі, тобто неінтерактивні елементи, яким виділяться більша половина екранного простору. Також наявні блоки «Додатковий простір», виділений для резервування вільного місця для додавання нових елементів, а графічна кнопка «Вихід» додана для виходу із третього стану ГІС до попереднього.

Другий стан ГІС, тобто «цифрова мапа» є найголовнішим за своїм значенням, отже для створення його МРЕІ потрібно приділити окрему увагу. Перш за все, представлення карти місцевості, так як ГІС показує дані про місцезнаходження станцій в тому числі на мапі, а споживачем є авто, що робить його окремою точною на карті, логічно буде застосувати схематичний вигляд місцевості з орієнтуванням на графічне виокремлення шляхів, які призначені для пересування авто – дороги. Тобто за виглядом мапи ГІС буде найбільше відповідати оформленню дорожньої карти. При цьому, для кращого сприйняття користувачем мапи будуть зведені до мінімуму сторонні позначення. Оскільки мапа надає основну інформацію про дані трекінгу користувачеві, взаємодія з нею має бути зручна не лише на рівні сприйняття умовних графічних позначок, а й на рівні сприйняття кольорів. Отже потрібно обирати тьмяні відтінки кольорів для заповнення великих площ на мапі, та яскраві кольори для позначок.

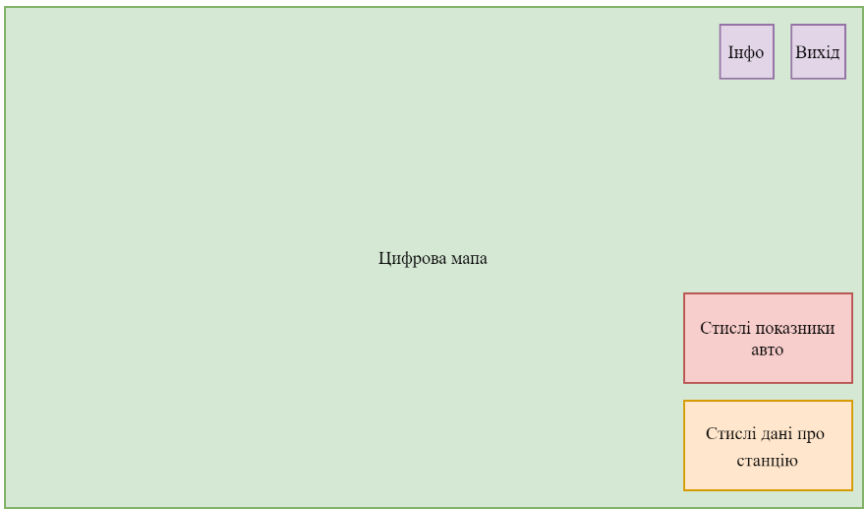

Рис. 3.13. Загальний вигляд МРЕІ для стану «Цифрова мапа»

На екрані має відображатися максимум картографічною візуальної інформації, тому наявність графічних кнопок або інформаційних панелей має бути мінімальна та розташовуватися ближче до країв екрану відображення. Для спрощення процесу розробки та зменшення кількості зайвої інформації на екрані будуть використовуватися власні графічні позначки географічних об’єктів у вигляді геометричних примітивів, які вирізнятимуться на фоні зображення загальної території, схематично зображеної на мапі.

Для відображення цифрової мапи використовується метод *drawMap()* класу *Map*. Цей метод відповідає за злагоджену роботу переміщення віртуальної камери, яка показує ту область, що бачить користувач на екрані. Клас віртуальної камери надає бібліотека *SFML*, цей клас має назву *View*. Рух віртуальної камери здійснюється через два основні методи: *setCenter()* та *setView()*.

Метод *setCenter()* відповідає за встановлення центральної точки та збереження її координат для подальшої обробки іншими методами.

Метод *setView()* поміщає уявну точку віртуальної камери на задані координати в двовимірному просторі. Так як користувач бачить ту ж область на екрані, що й віртуальна камера, метод *drawMap()* працює таким чином, щоб відображати мапу тільки тієї області, яку бачить користувач [13]. Це зумовлено тим, що мапа поділена на квадрати не лише для навігації, а й для візуалізації. Це означає, що при виводі зображення мапи, відбувається не виведення всієї мапи одразу, а певної множини квадратів, які разом складають область відображення тієї частини мапи, яку може охопити віртуальна камера за один кадр відтворення.

На рис. 3.14. схематично показано результат роботи методу *drawMap()*, де показано умовне зображення, поділене на квадратні фрагменти. Повне зображення тестової мапи з позначеннями точок джерел поповнення енергії у вигляді зарядних станцій для електромобілів представлене на рис. 3.15.

Використовуючи переміщення віртуальної камери через об’єкт *view* по зображенню мапи за допомогою клавіш-стрілок на клавіатурі та через графічні кнопки на екрані класу *Button*, з якими користувач взаємодіє через курсор, можливий огляд мапи.

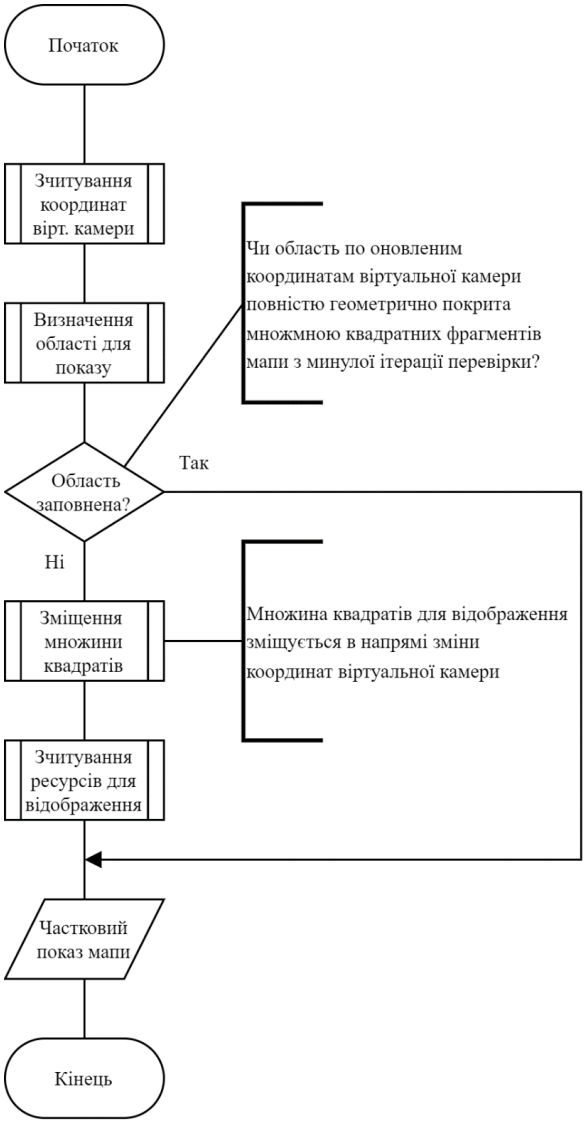

Рис. 3.14. Схема алгоритму роботи методу *drawMap()*

Рис. 3.15. Дорожня тестова мапа в числовому масштабі 1:200000

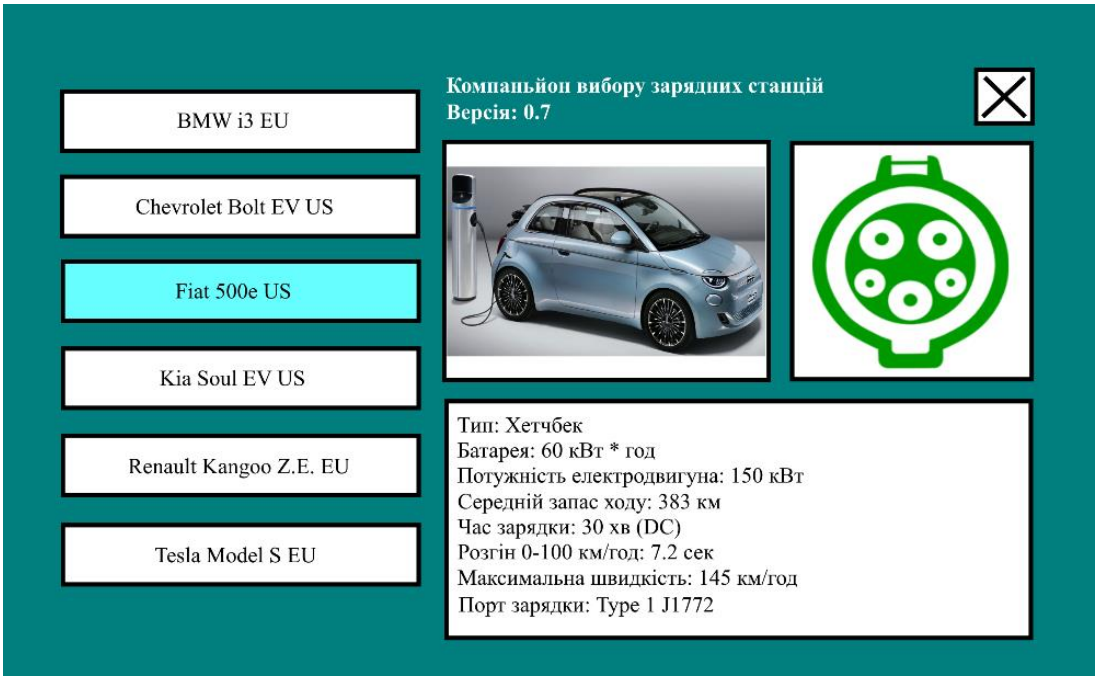

Рис. 3.16. Вікно програми для вибору авто

Рис. 3.17. Вікно програми для показу цифрової мапи

В результаті (рис. 3.16. - 3.17.) користувач може обирати підтримувану модель авто в переліку, та на основі технічних характеристик та даних трекінгу для джерел поповнення енергії та авто ГІС надає користувачеві дані про маршрут до рекомендованої станції, її зображення на мапі разом зі електрокаром-споживачем

та інформацію про станцію та авто на основі їх показників, які можуть бути прописані в самій ГІС або в якості даних з зовнішніх джерел.

3.5. Аналіз роботи розробленої геоінформаційної системи

Багато класичних концептуальних моделей [14, 15, 16] мають недостачу таких потрібних концепцій, як можливість багаторазового використання, ідентичності об’єкта, подання неточної інформації. А це вельми важливі речі при створенні деяких видів програм, як-от експертні системи баз даних (*EDS*), системи мультимедійних баз даних (*MMDBS*), системи автоматизації офісу (*OAS*), ну і звісно ж геоінформаційних систем (*GIS*) в тому числі. Така відсутність необхідних інструментів для створення концептуальних моделей ПЗ, подібних наведеним передбачено призвело до визначення більш потужних концептуальних моделей, враховуючи зростаючу складність.

Часто вживаними для створення ГІС-програм є концептуальні моделі *Enhanced Entity-Relationship*, *GraphDB*, *GODOT*. Вони основані на об’єктно-орієнтованому підході, так як він є більш простішим для розуміння та популярнішим при створенні ПЗ. Але одними з найновіших на момент проєктування є *Geo-ER*, та *GISER*, які в свою чергу є розширеннями моделі *Entity-Relationship* (*ER*).

Geo-ER – концептуальна модель, яка намагається зафіксувати особливості знаходження в просторі на рівні дизайну геоданих та їх впорядковування.

GISER – в свою чергу враховує чотири основотворні поняття:

– простір/час;

– особливість;

– охоплення;

– просторовий об’єкт.

Простір/час це в розумінні даної моделі безмежні багатовимірні простори, які мають власне положення та напрям та містять в собі різні географічні явища. Особливість – конкретне географічне явище (місто, дорога, територія, тощо).

Охоплення – це певна класифікація, сукупність, до якої можна віднести те чи інше географічний явище. Просторовий об’єкт – розглядається як підмножина першого поняття простір/час.

Деякі дослідники навіть зазначають [17], що концептуальні моделі загального призначення не підходять для їх застосування при проєктуванні ГІС. Тобто існує потреба у використанні підходящої концептуальної моделі для задоволення потенційних вимог, що можуть ставитися до геоінформаційної системи. Крім того, не варто забувати, що деякі специфічні властивості географічних або інших об’єктів системи, та і зв’язки між цими об’єктами можуть включати різні форми невизначеності.

Варто зазначити, що існує дві узагальнені моделі даних для представлення просторової інформації ГІС: польова та об’єктна моделі.

Польова модель представляє інформацію про простір на основі області на площині, яка має безперервний характер. До неї можуть належати різні просторові функцій. Така сукупність являє собою область атрибутів на площині. Кожна точка простору, яку займає об’єкт, має лише одне значення функції. Такі моделі застосовуються при позначенні явищ без чіткої форми: зон радіоактивного ураження, витоку певних газів, пожеж, або позначення неперервних просторових поверхонь (зміна температури, висоти, тиску тощо).

Об’єктно-орієнтовані моделі в свою чергу розглядають інформаційний простір як такий, що заповнений об’єктами, які можуть бути як просторовими, так і непросторовими, але можуть бути чітко розпізнаними. Такі об’єкти також є суцільними та прив’язаними до положення в просторі (транспортні мережі, об’єкти нерухомості, річки, земельні наділи тощо). В даному випадку межі знаходження таких об’єктів переважну більшість часу чітко сформовані, на відміну від польової моделі.

Просторові дані в ГІС мають складати мапу, яка має в собі набір визначених об’єктів, які відносяться до інформаційного простору. Географічні об’єкти можуть мати різні властивості, але пріоритетні для відтворення – саме геометричні, які

можна прив’язати до *2D*-геометрії, яка передбачає різні співвідношення точок та ліній на площині, а також можливість внесення деяких топологічних властивостей.

Взагалі, далеко не завжди можна точно описати всю семантику інформації навколишнього світу, так як наразі спостереження та фіксація об’єктів в ньому є все ж не досконалою, а як наслідок і відтворення, навіть в межах представлення є недостатнім. Саме поняття невизначеності може створюватись через самі є дані або зв’язки між об’єктами. А ГІС якраз є одна з тих типів програм, яка обробляє інформацію, яка в більшій мірі може неповною, неточною, або навіть місцями бути відсутня взагалі, тому необхідність врахування невизначеності через проєктування на основі концептуальних моделей зумовлене тим, що:

– деякі знання, які формалізуються та поміщаються в ГІС існують у вигляді «природньої мови», тобто створюються за допомогою численних описів (як-от «багато», «майже», «трохи», тощо) й нечітких термінів, так як інтуїтивно людині швидше виражати думки саме таким чином. Більшість таких описів нечіткі й використовуються саме для висловлення нечіткої інформації;

– вельми небагато природніх явищ можна описати чітко по межах, так як в більшості випадків маємо справу з «розмитим» варіантом переходу навіть говорячи про природні приклади, наприклад місця проживання диких тварин, схили, ґрунти типи рослинності. Ці явища мають неточні межі через їх перехідний характер.

Існують вимірювання, які в тому числі через свою прив’язку до просторової області часто є неповними. Якщо такі дані сприймати як абсолютно вірні, це може призвести до занадто великого відхилення, а то й до зовсім неправильних результатів.

Тож враховуючи вказані популярні концептуальні моделі, які використовують для створення ГІС-програм можна дійти думки, що ні одна з таких моделей не має справу з невизначеною інформацією на концептуальному рівні. Але існує потреба роботи і з просторовими об’єктами, і з обробкою специфічних невизначених властивостей. Тож варто було обрати концептуальну модель, яка зможе це задовольнити. Тож було прийнято рішення використання концептуальної моделі під назвою *ExIFO2*, якій під силу як зберігання набутих за довгий час

розробки старіших моделей сильні сторони семантичних підходів, так і приймання об’єктно-орієнтованих концепцій невизначеності [18].

Наукова новизна полягає у модернізації алгоритму підбору станції зарядки для електромобіля за даними трекінгу завдяки сучасному стандарту концептуальних моделей *ExIFO2*. Орієнтований граф, що створений під час проєктування, виконує спрощене графічне представлення семантики, дозволяючи на основі вершин створювати вже в програмному коді відповідні логічні конструкції через послідовність команд обраної мови програмування.

Практичне значення роботи полягає в тому, що результати, отримані в кваліфікаційній роботі використані в полегшенні процесу вибору джерела поповнення енергії (зарядної станції) по даним трекінгу відносно положення та характеристик авто (споживача) та візуалізації через *GUI*, що включає графічні кнопки, інформаційні панелі та цифрову мапу.

ГІС повністю виправдала очікування щодо вибору станції зарядки для електрокара через аналіз та візуалізацію даних трекінгу. Програма показує очікувані вірні результати при виборі моделі авто, показу цифрової мапи та додаткових інформаційних панелей на основі даних трекінгу джерел поповнення енергії.

3.6. Висновки до розділу

В третьому розділі кваліфікаційної роботи було описано процес розробки двох основних частин геоінфрмаційної системи: частини аналізу даних трекінгу та візуалізації даних. Для цього було обрано програмні засоби для розробки системи у вигляді мови програмування *C++* та інтегрованого середовища розробки *Microsoft Visual Studio 2017* в поєднанні з мультимедійною бібліотекою *SFML*.

Для створення програми на основі концептуальної моделі *ExIFO2* використано поділ життєвого циклу ГІС на програмні стани. В програмі присутні такі основні модулі, як модуль підтримки станів ГІС, модуль аналізу даних трекінгу та модуль візуалізації даних трекінгу.

В модулі підтримки станів ГІС використовуються такі основні класи, як *State* для створення станів та клас *AssetManager* для завантаження ресурсів зображень та шрифтів. Клас *State* містить методи для ініціалізації даних стану ГІС; для взаємодії з користувачем; для оновлення внутрішньої логіки ГІС; для відтворення графічних елементів ГІС, для призупинення активності стану без його видалення й для поновлення активності стану. Клас *AssetManager* містить методи для підготовки та завантаження ресурсів шрифтів та зображень до програми.

Для модуля аналізу даних трекінгу розроблено такі основні класи, як *Gobject, Car* та *Station*. Для класу *Gobject* розроблено метод для відображення графічних міток. Для класу *Car* створено методи для розрахунку додаткової інформації щодо показників авто. Для класу *Station* додано методи для виявлення територіальних квадратних зон на мапі, які не мають в собі географічних об’єктів станцій зарядки для електрокарів; для перевірки, чи знаходиться точка станції достатньо близько до точки місцезнаходження авто на мапі; для перевірки сумісності роз’єму електромобіля із зарядною станцією; для проведення додаткової перевірки, якщо в радіусі досяжності авто знаходиться декілька станцій, які пройшли основні перевірки та залишилися в переліку.

Для модуля візуалізації даних трекінгу розроблено основний клас *Map*, який містить в собі інформацію про положення та способи відображення об’єктів на мапі. Для класу *Map* також розроблено методи для часткового відтворення мапи згідно з положенням віртуальної камери, яка створена за допомогою класу *SFML View* та для ініціалізації початкових даних мапи.

При розробці ГІС було здобуто корисних навичок у проєктуванні структури програми за допомогою графу *ExIFO2*, розробці дизайну графічного інтерфейсу користувача, роботі з мультимедійною бібліотекою *SFML*, роботі з географічними даними, роботі зі стандартами режимів та типів зарядки електромобілів, створенні програмного коду на мові *C++*.

Робота виконана згідно діючого положення [19] та стандартів [20].

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було створено геоінформаційну систему аналізу та візуалізації даних джерел поповнення енергії, яка дозволяє допомогти користувачам електрокарів у пошуках станції для підзярядки авто з урахуванням його технічних характеристик. Крім аналізу даних трекінгу, на основі яких ГІС здійснює рекомендацію щодо станції для користування, система також надає користувачеві простий мінімалістичний інтерфейс для гнучкішої взаємодії з ГІС у вигляді ручного вибору моделі електрокару та демонстрування цифрової дорожної мапи для кращого відображення даних про просторове положення споживача та джерел поповнення енергії, також надаючи дані про географічні об’єкти у вигляді інформаційних панелей.

Було обгрунтовано значення геоінформаційних систем у сучасному світі, описані їх основні компоненти та обрано орієнтування програми на зберігання, аналіз та візуалізацію даних. Крім того був проаналізований програмний ринок ГІС. До уваги бралися різні аспекти схожих систем, особлива увага уділялася пошуку переваг та недоліків таких продуктів. Важливим також є інтуїтивно зрозумілий інтерфейс користувача в більшості програмних ГІС та можливість завантаження цифрових мап. Взято до уваги й мінімалізм панелей та меню керування, який подекуди зустрічався в *GUI* схожих програм. Вирішено, що візуалізацію треба проводити у *2D*-просторі задля простоти сприйняття користувачем та меншого витрачання апаратних ресурсів, так як в частини переглянутих аналогів графічний інтерфейс відчувається «перевантаженим», варто використати мінімалістичний підхід при створенні власного GUI; для використання ГІС не обов’язково перейматися про створення функції збору геоданих, так як їх можна створити/зібрати окремо самому, або зі сторонніх джерел; корисній системі необов’язково бути з громістким інструментарієм – варто зосередитися на основних функціях ГІС; для максимально комфортної взаємодії з ГІС користувачеві потрібна цифрова мапа; варто більше орієнтуватися на офлайн-

режим роботи ГІС, але в поєднанні з *GPS*; сенс розробляти систему є в першу чергу для портативних пристроїв.

Також було досліджено впровадження концепції підтримки нечітких даних до ГІС. Описано основні ідеї проєктування за допомогою концептуальних моделей та обрано конкретну модель для побудови системи. Для створення основи для подальшого проєктування та розробки проєкту, було здійснено проєктування системи за моделлю під назвою *ExIFO2*, яка зберігає сильні сторони семантичних підходів, так і приймання концепцій невизначеності з об'єктно-орієнтованою парадигмою. Це новий підхід до концептуальної побудови, що забезпечить можливість покращення подання нечітких даних, які властиві для даних, які часто обробляють ГІС. Використання такої моделі в основі ГІС ще на етапі проєктування забезпечує системі перспективи додавання стійкості до невизначеності та нечіткості початкових даних, які оброблятиме система. На основі *ExIFO2* була створена структура на основі графу основної частини ГІС-застосунку – цифрової мапи. Детально описано основний набір даних, який фігурує в ГІС – географічний об'єкт, названий *Gobject*, який складається з трьох основних типів: інформаційний опис, геодані та дані графічного відображення, які також описані та позначені на часткових графах. Було визначено та описано основні два алгоритми, які виконують поставлені перед системою задачі відповідають за аналіз та візуалізацію даних трекінгу джерел поповнення енергії, а саме:

– відбір найближчих до споживача станцій, яких може досягти споживач;

– визначення джерел поповнення енергії з характеристиками, що відповідають запитам споживача.

Крім того, були визначені компоненти для розробки геоінформаційної системи, такі як пакети для аналізу даних ГІС, статистичні пакети та пакети візуалізації даних до яких входить також описана в розділі мультимедійна бібліотека *SFML*. Описано процес розробки двох основних частин геоінфрмаційної системи: частини аналізу даних трекінгу та візуалізації даних. Для цього було обрано програмні засоби для розробки системи. Для створення станів ГІС використовувався клас *State*, який містить методи для підтримки станів:

– *start()*, для ініціалізації даних стану ГІС;

– *control()*, для взаємодії з користувачем;

– *update()*, для оновлення внутрішньої логіки ГІС;

– *show()*, для відтворення графічних елементів ГІС;

– *freeze()*, для призупинення активності стану без його видалення;

– *resume()*, для поновлення активності стану.

Крім того, було розроблено методи для аналізу даних трекінгу *isSquareEmpty()* для перевірки наявності в територіальних квадратах мапи наявність станцій, метод *isStationInR()* для перевірки чи знаходиться географічний об'єкт достатньо близько до споживача, метод *checkConnectors()*, що відповідає за перевірку роз'ємів, метод *add_check()* проводить додаткову перевірку в переліку *Candidates_st_list*, метод *warn_message()* для виведення попереджуючих повідомлень. Розроблено клас *Map* згідно з графом *ExIFO2*, метод *showPathArea()* візуалізує на мапі результати аналізу на основі даних трекінгу. Для візуалізації ГІС було розроблено класи *AssetManager* для завантаження ресурсів зображень та шрифтів до програми, *Button* для створення інтерактивних графічних кнопок, макети розташування елементів інтерфейсу (MPEI) для кращого розуміння при побудові візуальної частини всієї ГІС, а також розроблено метод *drawMap()* для часткового відтворення мапи згідно з положенням віртуальної камери.

При розробці ГІС було здобуто корисних навичок у проєктуванні структури програми за допомогою графів *ExIFO2*, розробці дизайну графічного інтерфейсу користувача, роботі з мультимедійною бібліотекою *SFML*, роботі з географічними даними, роботі зі стандартами режимів та типів зарядки електромобілів, створенні програмного коду на мові *C++*.

ГІС повністю виправдала очікування щодо вибору станції зарядки для електрокара через аналіз та візуалізацію даних трекінгу. Програма показує очікувані вірні результати при виборі моделі авто, показу цифрової мапи та додаткових інформаційних панелей на основі даних трекінгу джерел поповнення енергії.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке ГІС (геоінформаційні системи)? [Електронний ресурс]. – Режим доступу: *https://geoknigi.com/book_view.php?id=1456* (дата звернення 05.09.2022). – Назва з екрану.

2. Складові частини ГІС [Електронний ресурс]. – Режим доступу: *https://gis.kname.edu.ua/index.php/uk/2-uncategorised/67-skladovi-chastini-gis* (дата звернення 05.09.2022). – Назва з екрану.

3. Огляд сучасних програмних середовищ для розрахунків та аналізу даних [Електронний ресурс]. – Режим доступу: *https://vuzlit.com/629360/oglyad_suchanih_programnih_seredovisch_rozrahunkiv_analizu_danih* (дата звернення 06.09.2022). – Назва з екрану.

4. Класифікація сучасних ГІС [Електронний ресурс]. – Режим доступу: *https://geoknigi.com/book_view.php?id=641#:~:text=За%20призначенням%20геоінформаційні%20системи%20поділяють,однієї%20або%20кількох%20близьких%20функцій.* (дата звернення 19.09.2022). – Назва з екрану.

5. *IFO: a formal semantic database model* [Електронний ресурс]. – Режим доступу: *https://www.semanticscholar.org/paper/IFO%3A-a-formal-semantic-database-model-Abiteboul Hull/ea2e7b268c72cc6ab36a61f0b037b4553c91cf56* (дата звернення 01.10.2022). – Назва з екрану.

6. Science and technology as a basis of modernization for future sustainable development – SSF-2014: proceedings of International Humboldt Conference (18-21 September 2014, Minsk) / Editorial by Levchenko S., Kuzei A. – Minsk : 2014. – 112 p.

7. *Extending similarity-based fuzzy object-oriented data model* [Електронний ресурс]. – Режим доступу: *https://dl.acm.org/doi/epdf/10.1145/331119.331450* (дата звернення 03.10.2022). – Назва з екрану.

8. ЩО ПОТРІБНО ЗНАТИ ПРО GPS-ТРЕКІНГ [Електронний ресурс]. – Режим доступу: *https://skt-globus.com.ua/gps-tracking-chto-znat/* (дата звернення 03.10.2022). – Назва з екрану.

9. Всі види зарядок для електромобілів [Електронний ресурс]. – Режим доступу: *https://www.drom.com/info/misc/79745.html* (дата звернення 04.10.2022). – Назва з екрану.

10. Програмні засоби ГІС [Електронний ресурс]. – Режим доступу: *https://geoknigi.com/book_view.php?id=626* (дата звернення 04.10.2022). – Назва з екрану.

11. *SFML: Frequently Asked Questions (FAQ)* [Електронний ресурс]. – Режим доступу: *https://www.sfml-dev.org/faq.php* (дата звернення 22.10.2022). – Назва з екрану.

12. Уроки по графічній бібліотеці *SFML* [Електронний ресурс]. – Режим доступу: *https://ravesli.com/uroki-po-sfml/* (дата звернення 24.10.2022). – Назва з екрану.

13. Векторна алгебра в *SFML* [Електронний ресурс]. – Режим доступу: *https://ps-group.github.io/cxx/sfml_vector_math* (дата звернення 25.10.2022). – Назва з екрану.

14. *Hadzilacos, T. and N. Tryfona, "An Extended Entity-Relationship Model for Geographic Applications," SIGMOD Record, Vol.* 26, *No.*3, 1997, *pp:* 24-29.

15. *Worboy, M.F., "Object-Oriented Approaches to Geo-Referenced Information," Int. J. Geographical Information System*, Vol 8, *No.*4, 1994.

16. *Gunther O., Riekert W.F., "The Design of GODOT: An Object-Oriented Geographic Information System," IEEE Data Engineering Bulletin* 16, *No.*3, 1993.

17. *Shekhar, S., M. Coyle, B. Goyal, D. Liu, and S. Sarkar, "Data Models in Geographic Information Systems", Communication of ACM, Vol.* 40, *No.*4, *April* 1997, *pp:* 103-111.

18. *Conceptual Modeling of Geographic Information System Applications* [Електронний ресурс]. – Режим доступу: *https://www.researchgate.net/publication/228921801_Conceptual_Modeling_of_Geog*

raphic_Information_System_Applications (дата звернення 26.09.2022). – Назва з екрану.

19. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проєкти) випускників Національного авіаційного університету. – К.: НАУ, 2017. – 63 с.

20. ДСТУ ГОСТ 3008-95 «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення».