

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТРАНСПОРТНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут управління, технологій та правових наук
Кафедра інформаційних технологій

МАГІСТЕРСЬКА РОБОТА

на тему:

**ІНТЕЛЕКТУАЛЬНИЙ ПОМІЧНИК "CAPTAIN'S COMPANION" ДЛЯ
ГРАВЦІВ РОЗШИРЕННЯ НАСТІЛЬНО-РОЛЬОВОЇ ГРИ GURPS
SPACESHIPS**

Спеціальність: 121 «Інженерія програмного забезпечення»
ОПП «Інженерія програмного забезпечення»

Розробив: Михайло КАЧУРА
(ім'я, ПРІЗВИЩЕ)

(підпис)

Керівник: к.ф.-м.н., доцент Ольга ТКАЧЕНКО
(науковий ступінь, вчене звання) (ім'я, ПРІЗВИЩЕ)

(підпис)

Нормоконтроль: к.ф.-м.н., доцент Ольга ТКАЧЕНКО
(науковий ступінь, вчене звання) (ім'я, ПРІЗВИЩЕ)

(підпис)

Допущений до захисту
«_____» _____ 2025р.

В.о. завідувача кафедри: д.т.н., професор Олег НІКОНОВ
(науковий ступінь, вчене звання) (ім'я, ПРІЗВИЩЕ)

(підпис)

Рецензент: д.т.н., професор Андрій БОНДАРЧУК
(науковий ступінь, вчене звання) (ім'я, ПРІЗВИЩЕ)

(підпис)

Київ – 2025

Міністерство освіти і науки України
 Національний транспортний університет
 Навчально-науковий інститут управління, технологій та правових наук
 Кафедра інформаційних технологій

Рівень вищої освіти – другий (магістерський)
 Галузь знань 12 «Інформаційні технології»
 Спеціальність 121 «Інженерія програмного забезпечення»
 ОПП «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри
 інформаційних технологій
 д.т.н., проф. _____ Олег НІКОНОВ

«_____» _____ 2025 р.

Завдання
на магістерську роботу здобувачу вищої освіти
Михайлу КАЧУРІ

 (ім'я, прізвище)

1. Тема роботи: Інтелектуальний помічник "Captain's Companion" для гравців розширення настільно-рольової гри GURPS Spaceships

керівник роботи: к.ф.-м.н., доцент Ольга ТКАЧЕНКО
 Затверджені наказом від «17» жовтня 2025 року № 1031
 2. Строк подання здобувачем роботи 19 грудня 2025 року

3. Вихідні дані до роботи: Інтелектуальний помічник "Captain's Companion", база даних зоряних систем та космічних кораблів інтелектуального помічника "Captain's Companion"

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Програмна реалізація розрахунків орбітальних маневрів та моделювання космічних кораблів для настільно-рольової гри, створення інтелектуального помічника, проектування бази даних, опис предметної області, інсуючі програмні аналоги, створення графічного інтерфесу

5. Перелік графічного матеріалу: презентація магістерської роботи та скріншрти функціонування програмного продукту "Captain's Companion"

6. Консультанти роботи: к.ф.-м.н., доцент Ольга ТКАЧЕНКО

7. Календарний план виконання роботи

№ з/п	Найменування етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітки
1	Огляд та аналіз джерел тематичної області	17.10.25 – 22.10.25	
2	Огляд та аналіз шляхів розв’язання проблем роботи	23.10.25 – 28.10.25	
3	Опис власного підходу до вирішення проблеми	29.10.25 – 04.11.25	
4	Визначення вимог до програмного продукту та його структури	05.11.25 – 07.11.25	
5	Опис та реалізація основних функцій програмного продукту	08.11.25 – 28.11.25	
6	Оформлення пояснювальної записки	29.11.25 – 05.12.25	
7	Захист		

8. Дата видачі завдання «17» жовтня 2025 року

Здобувач вищої освіти _____ Качура М.О.
 (підпис) (прізвище та ініціали)

Керівник роботи _____ Ткаченко О.І.
 (підпис) (прізвище та ініціали)

РЕФЕРАТ

Тема: Інтелектуальний помічник "Captain's Companion" для гравців розширення настільно-рольової гри GURPS Spaceships

Магістерська робота: 153 с., 36 рис., 11 таблиць, 32 літературних джерела, 3 додатки.

Магістерська робота присвячена розробці інтелектуального помічника для гравців настільно-рольових ігор. Метою роботи є дослідження предметної області та розробка інтелектуального помічника для гравців настільно-рольових ігор з використанням розширення GURPS Spaceships.

Дослідження зосереджено на розробці інтелектуального помічника, що включає в себе функціонал для створення зоряних систем та космічних кораблів, виконання розрахунків орбітальних маневрів з можливістю зберігання інформації щодо зоряних систем та космічних кораблів для повторного використання. Застосовано такий стек технологій: мова програмування C#, мова запитів T-SQL та бібліотека Microsoft.Data.SqlClient для реалізації взаємодії інтелектуального помічника з базою даних.

Результатом роботи є розроблений інтелектуальний помічник для гравців настільно-рольових ігор з розширенням GURPS Spaceships, який дозволяє гравцям швидко створювати космічні кораблі за системою, описаною в розширенні, та виконувати розрахунки орбітальних маневрів, спрощуючи цей процес та зменшуючи кількість допущених помилок.

Створений інтелектуальний помічник має потенціал для подальшого розширення та вдосконалення, включаючи можливість роботи на віддаленому сервері, доступ декількох користувачів та збільшення опцій при створенні космічних кораблів, зоряних систем та видів орбітальних маневрів.

Ключові слова: інтелектуальний помічник, графічний інтерфейс, стек технологій, база даних, настільно-рольові ігри, орбітальні маневри, космічні кораблі.

ABSTRACT

Topic: Intelligent Assistant «Captain's Companion» for Players of the GURPS
Spaceships Role-Playing Game Expansion

Master's Thesis: 153 p., 36 figures, 11 tables, 32 references, 3 appendices.

The master's thesis is dedicated to the development of an intelligent assistant for tabletop role-playing game players. The aim of the work is to study the subject area and develop an intelligent assistant for tabletop role-playing game players that use the GURPS Spaceships expansion.

The research focuses on the development of an intelligent assistant that includes functionality for creating star systems and spacecraft, performing orbital maneuver calculations, and the ability to store information about star systems and spacecraft for reuse. The following technology stack was used: C# programming language, T-SQL query language, and the Microsoft.Data.SqlClient library to implement the interaction of the intelligent assistant with the database.

The result of the work is the development of an intelligent assistant for players of tabletop role-playing games with the GURPS Spaceships expansion, which allows players to quickly create spacecraft according to the system described in the expansion and perform orbital maneuver calculations, simplifying this process and reducing the number of mistakes made.

The created intelligent assistant has the potential for further expansion and improvement, including the possibility of operating on a remote server, multiple user access, and an increase in options when creating spacecraft, star systems, and types of orbital maneuvers.

Keywords: intelligent assistant, graphical interface, technology stack, database, tabletop role-playing games, orbital maneuvers, spacecraft.

ЗМІСТ

ВСТУП	7
1 ОГЛЯД ТА АНАЛІЗ ОБЛАСТІ ДОСЛІДЖЕННЯ – НАСТІЛЬНО-РОЛЬОВІ ІГРИ	10
1.1 Сутність області дослідження	10
1.1.1 Настільні та настільно-рольові ігри	10
1.1.2 Використання програмних помічників в настільних та настільно-рольових іграх	13
1.1.3 Система опису космічних кораблів	17
1.1.4 Орбітальні маневри	18
1.2 Існуючі помічники для настільної гри	20
1.3 Основні проблеми інтелектуалізації настільно-рольових ігор	27
2 АНАЛІЗ ШЛЯХІВ СТВОРЕННЯ ІНТЕЛЕКТУАЛЬНОГО ПОМІЧНИКА	30
2.1 Алгоритми розрахунку параметрів суб'єктів інтелектуального помічника	30
2.2 Архітектура інтелектуального помічника “Captain’s Companion”	40
2.3 Опис використаних технологій розробки інтелектуального помічника	46
3 ОПИС ПРОГРАМНОГО ПРОДУКТУ “CAPTAIN’S COMPANION”	50
3.1 Вимоги до інтелектуального помічника	50
3.2 Структура бази даних програмного продукту	52
3.3 Опис основних функцій інтелектуального помічника	67
ВИСНОВКИ	80
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	82
ДОДАТКИ	85

ВСТУП

В настільно-рольових іграх, як і в інших розвагах, важливим є як покращення ігрового процесу, так і зменшення часу, який витрачається не на розважальний процес, а на проміжні чи підготовчі дії.

Розвиток допоміжного програмного забезпечення (ПЗ), такого як програми-помічники, цифрові інструменти гравців, та т.і., спрямованого на автоматизацію, пришвидшення, спрощення проміжних дій гравців, та покращення ігрового процесу є актуальною проблемою в сфері настільно-рольових ігор.

Інтелектуальний помічник, створений для спрощення виконання складних чи довгих процесів під час гри, дозволить гравцям не лише швидше повернутись до цікавих їм елементів ігрового процесу, але й завдяки цьому спрощенню створить можливість багаторазового виконання складних процесів з різними параметрами, надаючи гравцям можливість краще порівняти декілька різних можливих дій та їхні результати.

Актуальність проблеми, що розглядається у даній магістерській роботі, полягає у тому, що наявні допоміжні інструменти для спрощення використання розширення настільної гри GURPS Spaceships мають обмежений функціонал, а правила цього розширення та пов'язаних ігрових матеріалів вимагають значних витрат часу гравців.

Науковою новизною роботи є використання інтелектуального помічника при вирішенні проблем забезпечення індивідуалізації та спрощення використання розширення настільно-рольової гри GURPS Spaceships.

Використання інтелектуального помічника, зокрема:

- збільшує ефективність гри всіх її учасників, підвищуючи рівень вмотивованості та зацікавленості грою через перекладання рутинних операцій розрахунків на інтелектуального помічника;
- зменшує час, необхідний гравцям для проведення розрахунків, обумовлених правилами гри.

Метою роботи є створення інтелектуального помічника для спрощення процесу створення космічних кораблів за системою GURPS Spaceships та виконання розрахунків орбітальних маневрів.

Для досягнення поставленої мети необхідно вирішити ряд завдань:

- зробити аналіз предметної області;
- визначити частини ігрового процесу, які потребують для свого виконання багато часу;
- проаналізувати процес створення космічних кораблів та виконання розрахунків орбітальних маневрів в настільно-рольових іграх;
- обґрунтувати створення програмного рішення проблеми;
- здійснити збір загальної інформації та експертних знань, необхідних для створення космічних кораблів та виконання розрахунків орбітальних маневрів;
- розробити структуру інтелектуального помічника;
- визначити вимоги до інтелектуального помічника, його функції та основні компоненти;
- розробити інтелектуальний помічник “Captain’s Companion”.

Об’єктом дослідження в магістерській роботі є застосування інтелектуальних помічників в настільних іграх.

Предметом дослідження в магістерській роботі є використання інтелектуальних помічників для спрощення та пришвидшення створення космічних кораблів за системою, описано в розширенні GURPS Spaceships та виконання розрахунків орбітальних маневрів.

Практична цінність магістерської роботи полягає у економії часу гравців та збільшенні задоволення від гри шляхом автоматизації виконання складних елементів ігрового процесу за допомогою інтелектуального помічника.

Магістерська робота складається зі вступу, трьох основних розділів, висновків, списку використаних джерел та застосунку, в яких розглядаються

предметна області настільно-рольових ігор, аналоги існуючих рішень, методи реалізації власного рішення, та його функціонал.

Особистий внесок магістра. Магістерська робота є самостійним завершеним дослідженням автора.

Публікації: дослідження були апробовані:

– Качура М.О., Овчарук І.В. Використання програмних застосунків як інтегральних елементів у настільних іграх. *Виклики та перспективи розвитку транспортної інфраструктури України: збірник тез III Всеукраїнської науково-практичної конференції* (м. Київ, 22 – 24 квітня, 2025). Київ: ДУІТ. С. 516-518. [10].

– результати дослідження опубліковані в науковій статті Качура М., Овчарук І. Настільно-рольові ігри та їхні комп'ютерні аналоги. *Norwegian Journal of development of the International Science*, 2025. № 151. С. 96-99. <https://doi.org/10.5281/zenodo.15536107> [2].

1 ОГЛЯД ТА АНАЛІЗ ОБЛАСТІ ДОСЛІДЖЕННЯ – НАСТІЛЬНО-РОЛЬОВІ ІГРИ

1.1 Сутність області дослідження

1.1.1 Настільні та настільно-рольові ігри

Обрана тематична область стосується використання ПЗ в настільно-рольових іграх. Настільні ігри включають в себе велику кількість категорій. Настільно-рольові ігри є однією з категорій настільних ігор та можуть поділятися на наративні ігри та ігри-симуляції.

Наративні ігри характеризуються менш жорсткими правилами та більшою свободою дій гравців, а ігри-симуляції мають чіткі правила на більшість ігрових ситуацій. Ігри-симуляції не обмежують дії гравців набагато сильніше за наративні ігри, але ці дії повинні відповідати вимогам правил, якщо такі наявні для певної ситуації.

Важливо зауважити, що це залежить не лише від самої гри, але й від домовленості між гравцями та майстром гри про очікуваний стиль гри, але деякі ігрові системи спроектовані для певного стилю гри, тому краще відповідатимуть очікуванням гравців.

Для порівняння таких підходів можна навести приклади бойової ситуації для двох настільно рольових ігор – D&D (Dungeons & Dragons) та GURPS. П'яте видання гри D&D має такий алгоритм дій під час бою:

1. Гравець обирає ціль та кидає кубик для атаки, до якого додаються відповідні атрибути персонажа.
2. Результат кидка кубика порівнюється з класом броні цілі, якщо результат більший чи дорівнює класу броні, атака влучає.

3. Якщо атака влучає, гравець кидає кубик пошкоджень зброї свого персонажа та додає атрибути персонажа, які впливають на кількість пошкоджень.

4. Пошкодження віднімаються від очок життя цілі.

5. Якщо цілі нанесено пошкодження, які дорівнюють чи більше за половину її очок життя, ціль отримує додатковий негативний ефект.

6. Якщо ціль має 0 чи менше очок життя, вона втрачає свідомість або помирає.

Цей алгоритм може модифікуватись в залежності від класу персонажа який здійснює атаку, але є загальним прикладом атак в цій грі.

В четвертому виданні настільно-рольової гри GURPS, повна назва якої Generic Universal RolePlaying System, порядок вирішення результатів атаки має інший вигляд, який додатково розширений в додаткових матеріалах, таких як GURPS Martial Arts [1]:

1. Гравець обирає ціль, тип атаки, локацію атаки, та техніку атаки, кидає кубики та порівнює результат з рівнем вміння персонажа, додаючи до рівня бонуси чи штрафи за обрані ціль, тип, локацію, техніку атаки, та події минулого ходу якщо результат менший чи дорівнює ефективному рівню вміння, атака влучає.

2. Ціль обирає тип захисту та вирішує чи буде виконувати додаткові захисні дії, кидає кубики та порівнює результат з рівнем захисту, додаючи до рівня бонуси та штрафи за додаткові дії, тип, локацію, техніку атаки, та події минулого ходу, якщо результат менший чи дорівнює ефективному рівню захисту, атака відхилена.

3. Якщо атака не відхилена та обрана локація містить підлокацію, виконується кидок для визначення влучання в підлокацію.

4. Гравець кидає кубики пошкоджень для зброї та віднімає значення броні цілі в обраній локації чи підлокації, яке ділиться на дільник броні, якщо зброя такий має. Якщо значення пошкоджень більше за модифіковане значення броні, атака пробиває броню, очки пошкодження, які пробили

броню, множаться на модифікатор пошкоджень, який залежить від зброї та типу атаки, і який також модифікується в залежності від локації чи підлокації куди влучила атака. Пошкодження додатково обмежуються максимальним значенням, якщо локація має таке обмеження.

5. Кількість пошкоджень віднімається від очок життя персонажа, що було нанесено достатньо пошкоджень та ціллю була кінцівка, кінцівка виведена з ладу чи знищена.

6. Якщо нанесені пошкодження більші чи дорівнюють половині очок життя цілі, було виведено з ладу чи знищено кінцівку, або було нанесено пошкодження локації чи підлокації, яка до цього призводить, ціль кидає кубики та порівнює результат з модифікованим значенням здоров'я. Якщо результат кидка більший за це значення, ціль падає або втрачає свідомість.

7. Якщо ціль сумарно отримала двократну кількість пошкоджень відносно очок життя, відбувається кидок кубиків, який порівнюється з модифікованим значенням здоров'я. Якщо результат більший за це значення, ціль помирає. Аналогічна перевірка відбувається при отриманні сумарних пошкоджень кратних трьом, чотирьом та п'ятьом. При досягненні шестикратною сумарною кількістю пошкоджень, ціль помирає автоматично.

Порівнюючи ці правила, GURPS має набагато більш деталізовану бойову систему, тому на шкалі наративності та симуляції є більш наближеним до симуляції в порівнянні з D&D.

Водночас, більшість з цих правил можуть бути проігноровані чи спрощені, якщо у гравців та майстра гри є бажання проводити простішу гру, бо при використанні більшої кількості правил для визначення результатів дій гравців, ігри-симуляції вимагають більшої кількості часу для виконання цих дій [3].

1.1.2 Використання програмних помічників в настільних та настільно-рольових іграх

В залежності від особливостей категорії та конкретної гри, в ігровому процесі може використовуватись ПЗ різної складності та різного рівня інтеграції в гру. При використанні ПЗ в настільних іграх загалом, воно може відігравати як допоміжну роль, так і бути невід'ємною частиною ігрового процесу. До допоміжного ПЗ відносяться програми, використання яких можна уникнути, не унеможливлюючи при цьому гру, такі як програмні помічники. Прикладом такого помічника є віртуальні гральні кубики. Цей функціонал може бути реалізований у вигляді веб сайтів [3] чи завантажуваних програм [4], або ж бути реалізованим в межах інших систем.

Більш спеціалізованим прикладом такого ПЗ є Munchkin Level Counter [5]. Це ПЗ призначене для мобільних пристроїв та може використовуватись гравцями ігор серії Munchkin [6] для підрахунку рівнів, кидків кубика, та отримання одноразової переваги під час гри. Таким чином це ПЗ не лише спрощує ігровий процес, але й розширює його, надаючи гравцям з застосунком додаткові можливості.

Прикладом настільних ігор які вимагають ПЗ для функціонування може бути настільна гра XCOM: The Board Game [7, 8]. Для цієї гри гравцям необхідно мати додатковий застосунок XCOM: TBG [9], який сповіщає їх про ігрові події, визначає порядок ходів гравців, який кожного раунду є випадковим, обмежує в часі хід гравців за допомогою таймера, дозволяє вибрати один з трьох рівнів складності, навчає гравців правилам, та реалізовує інший функціонал, необхідний для ігрового процесу [8].

Перевагою такої інтеграції ПЗ в настільні ігри є можливість розширити ігровий процес та механіки за межі звичайних настільних ігор, передаючи ПЗ виконання дій, які є або дуже складними для гравців, як розрахунок формул, монотонними, як кидання кубиків та перевірка результатів з декількома таблицями результатів, або ж занадто довгими для ручного виконання,

дозволяючи настільним іграм досягти ігро-механічної складності, подібної до комп'ютерних ігор [10].

Основним недоліком такого підходу є доступність ПЗ. Для наведеного прикладу, XCOM: The Board Game, сторінка настільної гри на сайті виробника [7] має п'ять посилань на завантаження програми для різних операційних систем, одне з яких, посилання на Play Market, виводить помилку при спробі відкриття, пов'язану з відсутністю програми в Play Market.

Існують альтернативні методи отримання застосунку для мобільних пристроїв з операційною системою Android, але відсутність застосунку за одним посиланням може означати припинення підтримки застосунку розробником, що включає припинення його адаптації до новіших версій операційних систем.

З часом, якщо такий застосунок не буде доступний для завантаження чи роботи на наявних у користувачів пристроях, можливість грати в таку настільну гру зникне.

Ще одним прикладом використання ПЗ в настільних іграх є Monopoly Super Electronic Banking [11]. Ця гра, та інші ігри серії з електронним обліком грошей [12], дозволяють гравцям використовувати ігрові картки, за якими закріплюється кількість грошей на рахунку гравців.

Окрім цього основного функціоналу, різні версії ігор мали додаткові можливості, як сканування карток нерухомості та їхній електронний облік [12]. Відмінністю інтеграції ПЗ цієї гри від попереднього прикладу є наявність спеціального пристрою в комплекті настільної гри. Це також має свої переваги та недоліки.

Перевагами є відсутність необхідності встановлення додаткового ПЗ на власний пристрій та пов'язана з цим можливість відсутності доступу до ПЗ чи підтримки його пристроями гравців.

Основним недоліком такого підходу є залежність гри від пристрою, який виробляється в обмеженій кількості та включає в себе малорозповсюджене обладнання, як зчитувач карток. При поломці такого пристрою його ремонт

буде складнішим через його рідкість, а емуляція віртуального середовища для роботи інтегрованого ПЗ є складною та непрактичною для переважної більшості гравців через специфічність цього середовища, відсутності ПЗ у вільному доступі, та необхідність взаємодії ПЗ з елементами пристрою, які взаємодіють з фізичними об'єктами, як зчитувач карток.

Для настільно-рольових ігор додаткове ПЗ, переважно програми-помічники (в тому числі й інтелектуальні помічники) також може застосовуватись. Одним з видів такого ПЗ є віртуальний лист персонажа.

Фізичний лист персонажа це один чи декілька листів, які описують персонажа, його здібності, речі, та інші важливі характеристики. Зазвичай такі листи мають певну стандартизовану форму для кожної гри, зумовлену використанням однакового переліку атрибутів для всіх персонажів, які створюються в межах гри.

Фізичний лист персонажа має певні недоліки, які використання програмного помічника дозволяє усунути, а саме зношуваність та можливість втрачання. Зношування листа відбувається при коригуванні інформації шляхом стирання попередніх записів на листі, або ж при випадковому пошкодженні листа при його перенесенні чи зберіганні.

Окрім остаточної втрати фізичного листа персонажа, він може бути залишений гравцем вдома при проведенні гри в іншому місці, унеможливаючи доступ до нього під час гри.

Використання електронного листа персонажа дозволяє його зберігати на пристрої гравця [13] чи в хмарному сховищі при використанні одного з онлайн-ресурсів для ведення листа персонажа [14]. Це робить втрату чи пошкодження листа менш вірогідним та забезпечує кращу доступність листа персонажа і спрощує його переміщення.

Ще одним популярним видом програмного помічника для настільно-рольових ігор є віртуальний ігровий стіл [15]. Віртуальний ігровий стіл зазвичай використовується під час бойових сегментів гри для відображення ігрової мапи та позицій персонажів. Ця програма-помічник може

функціонувати в межах одного пристрою, локально, але більш розповсюдженим його використанням є доступ через мережу декількох гравців.

Такий функціонал є корисним при проведенні настільно-рольової гри в онлайн-режимі, коли гравці не можуть бути фізично присутніми та спілкуються за допомогою мобільних пристроїв чи персональних комп'ютерів.

Використання такого програмного помічника покращує сприйняття гравцями ігрової ситуації, дозволяючи їм краще та швидше приймати рішення щодо дій під час їхнього ходу. При використанні віртуального ігрового столу локально, зображення зазвичай виводиться на спільний екран для всіх гравців.

Такий підхід дозволяє використовувати ігрові мапи без необхідності друку та застосовувати спеціальні ефекти, як рухомі фонові елементи мапи, що на паперових мапах зробити неможливо. Віртуальні ігрові столи також можуть забезпечувати додатковий функціонал, як інтеграцію листів персонажа чи можливість здійснювати кидки кубиків, що дозволяє пришвидшити ігровий процес, усуваючи такі дії гравців як пошук кубиків чи підрахунок результатів [2].

Ці види програм-помічників можуть використовуватись як для наративних ігор, так і для ігор які схилиються до симуляцій. Для ігор-симуляцій корисними є програмні помічники, які спрощують складні чи часто повторювані елементи ігрового процесу.

Прикладом такої програми-помічника є GURPS Vehicle Builder [16], призначений для створення транспортних засобів за правилами, наведеними в книжці GURPS Vehicles [17].

Ці правила призначені для створення транспортних засобів за межами ігрових сесій, але є доволі складними, вимагаючи великої кількості підрахунків та урахування особливостей обраних для транспортного засобу систем, що й призвело до створення програми-помічника. Його інтерфейс є простим при використанні, але користувач все ще повинен мати базові знання

щодо функціонування правил створення транспортних засобів GURPS Vehicles [17].

1.1.3 Система опису космічних кораблів

Основою системи опису космічних кораблів, наведеної в GURPS Spaceships [18], є 20 можливих слотів для встановлення систем космічного корабля, які забезпечують його функціонування. Всі системи, описані в GURPS Spaceships, відповідають певному розміру космічного корабля, відомому як “модифікатор розміру”, спрощуючи вибір необхідних систем та їхніх характеристик.

В залежності від модифікатору розміру космічного корабля змінюються його очки життя [19], довжина, рейтинг маневреності та рейтинг стабільності [20]. Окрім цих характеристик, модифікатор розміру впливає на доступність деяких систем, вартість та кількість екіпажу для кожної встановленої системи, та інші характеристики, які залежать від розміру та є специфічними для окремих систем. В GURPS Spaceships 4: Fighters, Carriers, and Mecha [21] було розширено доступні модифікатори розміру, з правилами використання менших космічних кораблів та адаптації існуючих систем під цей розмір.

Додатково до 20 доступних слотів, правилами передбачені такі ознаки космічного корабля як аеродинамічність, наявність крил, обертальної гравітації, маскування та динамічного візуального маскування.

Важливими елементами космічного корабля є рівень автоматизації космічного корабля [18; 22], який впливає на кількість екіпажу та вартість космічного корабля, та якість космічного корабля [23], яка впливає на вартість та надійність космічного корабля.

При заповненні двадцяти систем та інших характеристик космічного корабля можливо вирахувати його похідні характеристики, такі як рівень надійності, фактичні рейтинги маневреності та стабільності, атмосферну

швидкість, прискорення, запас палива та його вартість, та характеристики озброєння [18; 24].

1.1.4 Орбітальні маневри

Для виконання розрахунків орбітальних маневрів розглядаються три види трансферів, призначених для зміни орбіти космічного корабля навколо зірки: гоманівський трансфер, біеліптичний трансфер, та брахістохронний трансфер [25].

Гоманівський трансфер полягає в досягненні еліптичної орбіти за допомогою першого прискорення, а після виходу на необхідний орбітальний радіус, використання другого прискорення для його закріплення (рис. 1.1). Перевагою вибору такого трансферу є його ефективність у використанні палива, що дозволяє досягти більш віддалених орбіт.

Хоча ця траєкторія польоту є більш ефективною, вона має важливий недолік, порівняно з іншими траєкторіями, які покладаються на використання більшої кількості палива – це швидкість досягнення бажаної орбіти, яка в такому випадку є меншою, вимагаючи більше часу.

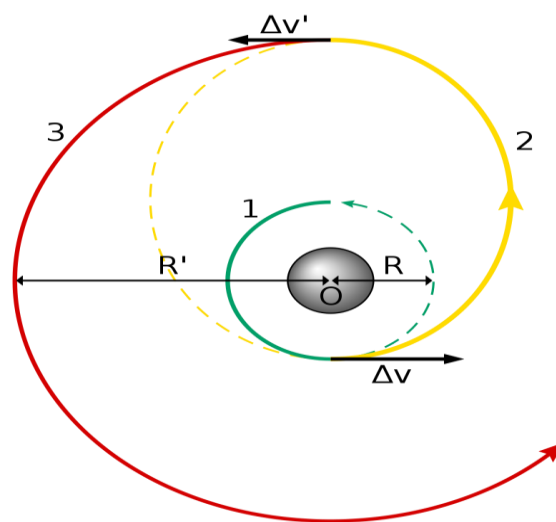


Рисунок 1.1 – Гоманівська траєкторія

В залежності від радіусу фінальної орбіти, більш ефективною у використанні палива може бути біеліптичний трансфер. Він є більш ефективним, якщо половина більшої осі початкової та фінальної орбіт відрізняються в 12 разів чи більше. Окрім більшої ефективності використання палива, така траєкторія має певні недоліки, наприклад набагато довший час, необхідний для досягнення бажаної орбіти.

Цей трансфер потребує трьох прискорень. Перше прискорення досягає еліптичної орбіти, після чого друге корегує її в іншу еліптичну орбіту, яка проходить через бажану орбіту. Коли космічний корабель досягає її, третє прискорення стабілізує його на новій орбіті (рис. 1.2). Чим більша відстань від зірки, досягнута внаслідок першого прискорення, тим більше палива буде заощаджено, але ціною часу, необхідного на виконання маневру.

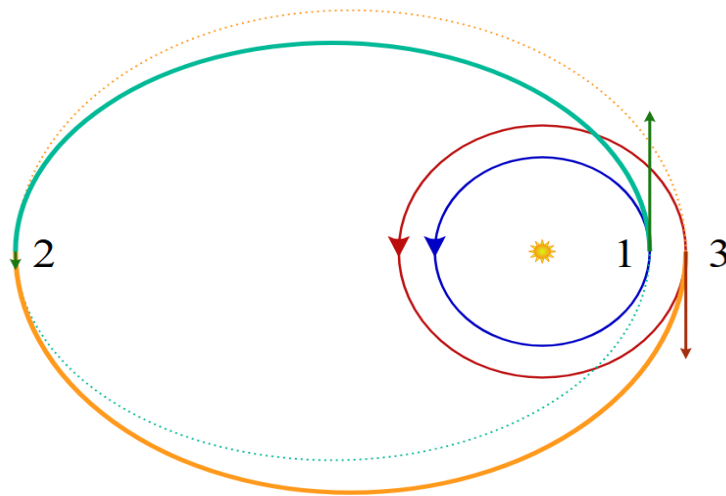


Рисунок 1.2 – Біеліптична траєкторія

Брахістохронна траєкторія, на відміну від попередніх, дозволяє мінімізувати час, необхідний для виконання маневру за рахунок використання великої кількості палива.

Для оптимального виконання маневру необхідно мати можливість постійно прискорюватись під час перельоту, що вимагає ефективних двигунів.

Цей метод полягає в постійному прискоренні до цільової орбіти, а після досягнення певної точки, прискорення в зворотному напрямку, вирівнюючи вектор польоту з необхідною орбітою.

Додатково до постійного прискорення з урахуванням необхідності зміни орбіти, можливий варіант прискорення лише під час певної частини маршруту, призначений для спрощених розрахунків при перельоті між двома точками, який наведений в книжці GURPS Spaceships [18].

1.2 Існуючі помічники для настільної гри

Так як розроблюваний інтелектуальний помічник суміщає декілька функцій, прямого аналогу цього ПЗ не існує. Натомість при розробці інтелектуального помічника було розглянуто аналоги реалізації окремих його функцій.

Для частини інтелектуального помічника, яка відповідає за створення космічних кораблів, було розглянуто два аналоги. Перший аналог – це Excel-файл “Spaceships v2.0” [26] (рис. 1.3).

Цей файл дозволяє створити космічний корабель, використовуючи правила GURPS Spaceships, додаткові правила, описані у випусках журналу Pyramid [27-29], та створені автором чи гравцями опціональні правила, які описані в розділі “Optional Rules” файлу та pdf-файлі, який додається при завантаженні.

Окрім створення космічного корабля, цей файл дозволяє враховувати такі фактори як яскравість та відстань до зірки при використанні сонячних вітрил, кількість необхідного екіпажу, розділену по ролям, необхідну кількість запасів їжі, та вартість виплат по кредиту за космічний корабель, описану в GURPS Spaceships 2: Traders, Liners, and Transports [23].

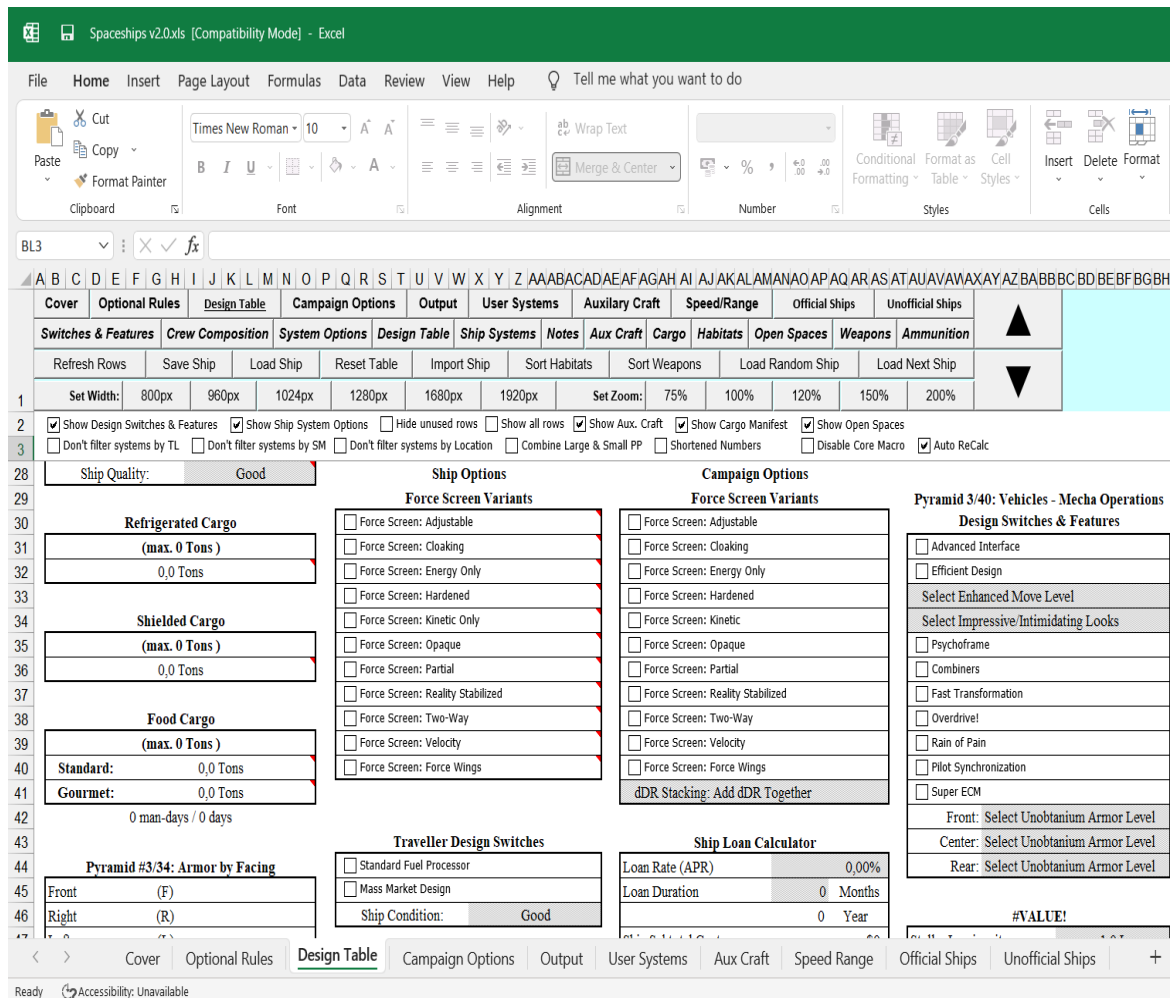


Рисунок 1.3 – Excel-файл “Spaceships v2.0”

Додатково до існуючих систем, користувач може створити власні системи в розділі “User Systems” для подальшого використання. Цей файл також містить готові космічні кораблі (офіційні та неофіційні), які користувач може редагувати.

Основним недоліком цього аналогу є його вимога до специфічної версії ПЗ для коректної роботи, а саме Microsoft Excel 2000, 2007, чи 2010, або ж OpenOffice 3.3.0. Враховуючи застарілість цього ПЗ, більшості користувачів необхідно буде встановити ці версії ПЗ, так як при відкритті файлу за допомогою Microsoft Excel 2021, критична частина функціоналу не працювала коректно. При спробі відкрити файл за допомогою Google Sheets, більша частина функціоналу файлу також не працювала, обмежуючи можливість користувача при роботі з хмарним сховищем. Ще одним недоліком є велика

кількість елементів керування, доступ до яких ускладнюється через наявність трьох закріплених рядків таблиці, які залишаються при навігації в нижню частину файлу, чим зменшується видима робоча область.

Серед обраних автором підходів при реалізації власного рішення було відмічено використання випадających списків, що спрощувало вибір між декількома варіантами певних опцій та буде впроваджено у власному інтелектуальному помічнику при створенні космічних кораблів.

Другим розглянутим прикладом серед існуючих рішень був вебзастосунок GURPS Spaceships Construction Program [30]. Цей застосунок має меншу кількість доступних опцій для створення космічних кораблів ніж попередній приклад, але спрощений процес може бути корисним для початківців чи користувачів, для яких не є важливим повний обсяг деталей (рис. 1.4).

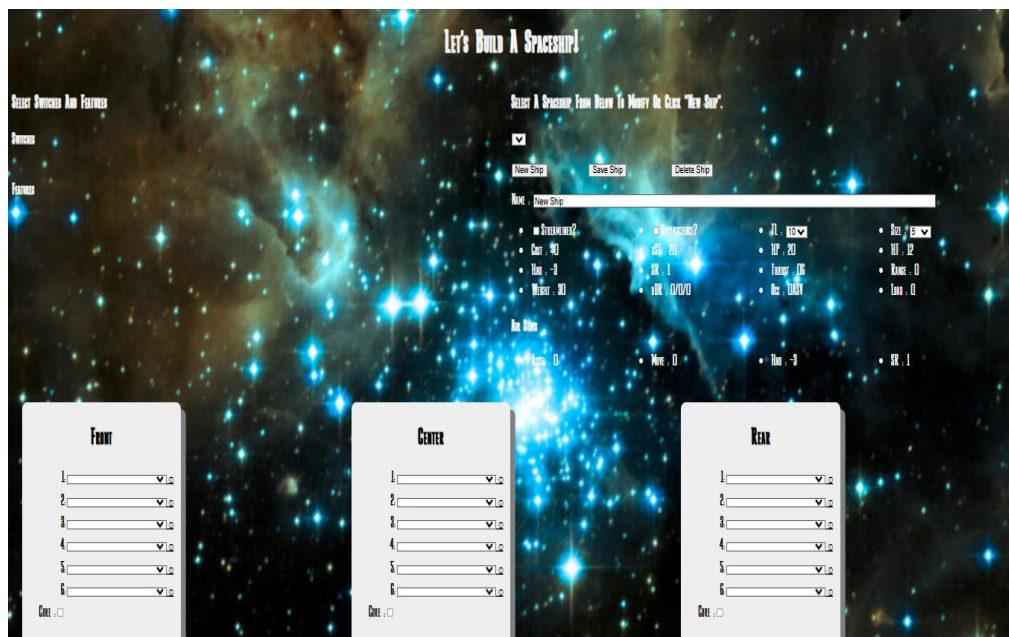


Рисунок 1.4 – Інтерфейс GURPS Spaceships Construction Program

Серед елементів керування доступних користувачам є випадające списки, призначені для вибору більшості характеристик космічного корабля, текстове поле для введення його назви, два поля з відмітками, які відповідають за форму корабля та використання нереалістичних технологій, та кнопки для

створення, видалення, та збереження космічного корабля. При спробі взаємодії з вебзастосунком GURPS Spaceships Construction Program, більшість випадуючих списків були порожні, що спричинено виникненням великої кількості помилок, які було виявлено при огляді сторінки в режимі розробника.

Такий аналог наразі не є працездатним, але демонструє можливі недоліки при розробці вебзастосунків – відсутність доступу до необхідних ресурсів чи застарівання технологій, які використовувались під час розробки. Розгляд цього аналогу призвів до прийняття рішення стосовно розробки локальної версії інтелектуального помічника.

Для частини інтелектуального помічника, пов'язаної з розрахунками палива та часу під час космічних перельотів було розглянуто офіційний Excel-файл, який надається в комплекті при придбанні випуску журналу Pyramid #3/79: Space Atlas [25].

В цьому файлі користувач може розрахувати витрачені паливо та час для перельотів за трьома варіантами трансферу, наведених в статті “Halfway to Anywhere” [25], та зльоту з поверхні планети.

Окрім цього, в файлі передбачено розрахунок маневрів при різниці в орбітальних площинах точок відправлення та призначення, та використання інших планет для отримання додаткового прискорення при короткочасному заході на їхню орбіту.

На відміну від файлу, розглянутого для побудови космічних кораблів, цей файл успішно відкривається в сучасних версіях ПЗ та онлайн редакторах, таких як Google Sheets.

Зовнішній вигляд файлу доволі простий для зручності роботи користувача (рис. 1.5), дозволяючи швидко навчитись ним користуватись.

\$10

fx

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																
1																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
2		Bi-Elliptic Transfer Calculations Support for <i>Halfway to Anywhere</i> from <i>Pyramid #3/79: Space Atlas</i>																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
3																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
4																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
5		Please see pp. 25-33 of Pyramid #3/79 for detailed information about how to fill in the values for this spreadsheet.																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
6		Additional worksheets for other calculations are in different worksheet tabs (below).																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
7																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
8		Provide information in the yellow cells. Units of measure are adjustable with their dropdown boxes.																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
9																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
10		Input															SI																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																

Рисунок 1.5 – файл “Pyramid #3-79 - Halfway to Anywhere Delta-V Spreadsheet”

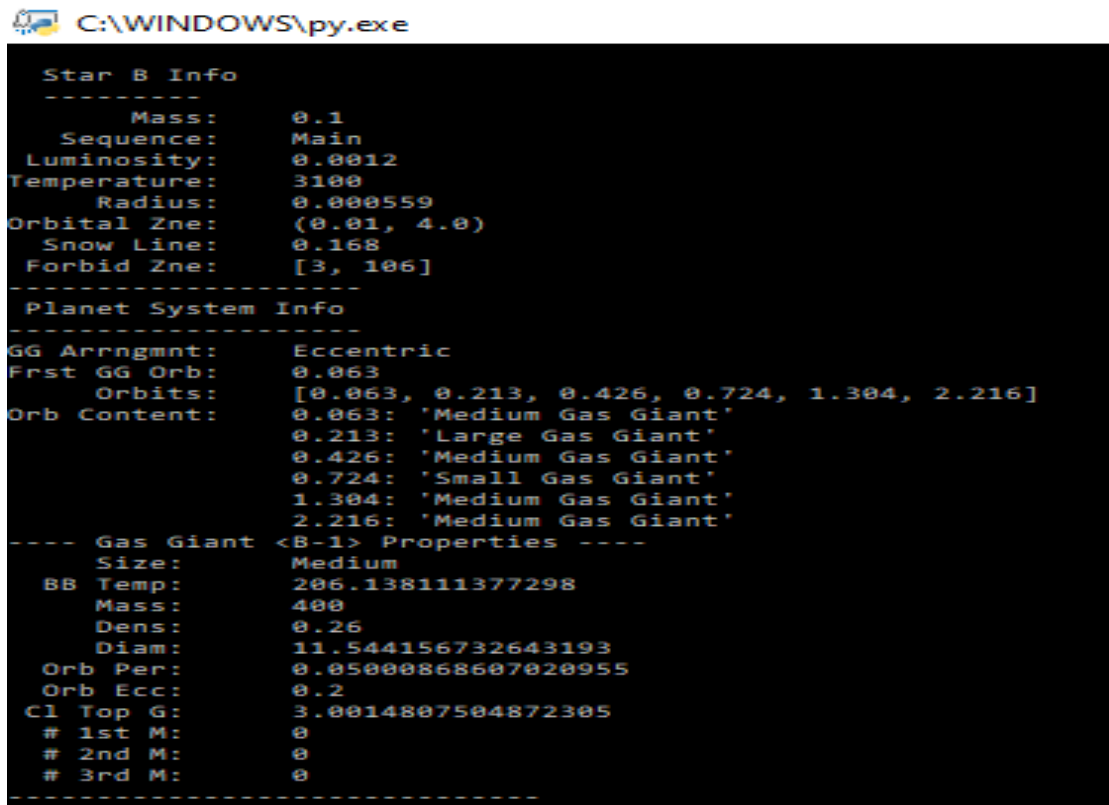
При аналізі цього файлу було поглиблене розуміння функціонування формул, наведених в статті “Halfway to Anywhere” [25] та помічено невідповідність вказаних одиниць вимірювання в одній з наданих формул, яку було виправлено при реалізації функціоналу розрахунків орбітальних маневрів інтелектуального помічника.

Для частини інтелектуального помічника, яка відповідає за створення зоряних систем, в яких відбуваються маневри, було розглянуто GURPS 4th Edition Star System Generator [31].

Це ПЗ дозволяє користувачу згенерувати випадкову зоряну систему за правилами генерації, описаними в книжці GURPS Space [32]. Так як це ПЗ створене за допомогою мови програмування Python, воно потребує встановлення інтерпретатора для функціонування, але не є вибагливим до операційної системи комп’ютера.

Як свідчить сторінка [31], на якій воно розміщене, це ПЗ не є повністю розробленим, а останні зміни в код вносились в 2020 році. Враховуючи це, майбутнє доопрацювання цього ПЗ є маловірогідним. При розгляді наявного функціоналу було помічено певні недоліки.

Один з недоліків, пов'язаний з використанням мови Python, це необхідність завантаження додаткових модулів. Сторінка ПЗ містить інструкцію для користувачів, але вона вимагає використання командного рядка, що не є зручним для звичайного користувача. Взаємодіяти з цим ПЗ користувач може за допомогою або веб-інтерфейсу, або командного рядка (рис. 1.6).



```

C:\WINDOWS\py.exe
Star B Info
-----
      Mass:      0.1
  Sequence:    Main
Luminosity:    0.0012
Temperature:   3100
      Radius:    0.000559
Orbital Zne:   (0.01, 4.0)
    Snow Line:  0.168
   Forbid Zne:  [3, 106]
-----
Planet System Info
-----
GG Arrngmnt:   Eccentric
  Frst GG Orb: 0.063
      Orbits:  [0.063, 0.213, 0.426, 0.724, 1.304, 2.216]
Orb Content:   0.063: 'Medium Gas Giant'
               0.213: 'Large Gas Giant'
               0.426: 'Medium Gas Giant'
               0.724: 'Small Gas Giant'
               1.304: 'Medium Gas Giant'
               2.216: 'Medium Gas Giant'
---- Gas Giant <B-1> Properties ----
      Size:     Medium
    BB Temp:    206.138111377298
      Mass:     400
      Dens:     0.26
      Diam:     11.544156732643193
    Orb Per:    0.05000868607020955
    Orb Ecc:    0.2
    Cl Top G:   3.0014807504872305
    # 1st M:    0
    # 2nd M:    0
    # 3rd M:    0
-----

```

Рисунок 1.6 – Приклад консольного інтерфейсу

При використанні командного рядка, інформація про створену зоряну систему виводиться у файл з розширенням “.tex”, який вимагає спеціалізованого ПЗ для перегляду. Інформація також виводиться у вікно командного рядка, яке закривається відразу після запуску, чому можна

запобігти внесенням команди паузи в код програми, але це рішення не є задовільним з точки зору звичайного користувача.

Запуск веб-інтерфейсу також вимагає використання командного рядка та навігації в кореневу директорію ПЗ, що аналогічно не є зручним для користувача. ПЗ дозволяє внесення змін в процес генерації зоряної системи, але це вимагає зміни параметрів всередині одного з файлів ПЗ. Після розгляду цього ПЗ, було прийняте рішення про мінімізацію залежності працездатності власного інтелектуального помічника від дій користувача, які напряду не пов'язані з роботою з ним. У випадку якщо такі дії необхідні, вони повинні бути максимально простими.

Ще одним прикладом ПЗ, яке призначене для створення зоряних систем є “Protodisc” [33] (рис. 1.7). Це ПЗ дозволяє користувачу швидко згенерувати випадкову зоряну систему за правилами генерації GURPS Space [32].

Перевагою цього ПЗ є наявність функціонуючого візуального інтерфейсу та деталізація генерації зоряних систем та їхніх елементів, яка значно перевищує мінімальну необхідну кількість даних, необхідних для виконання розрахунків орбітальних маневрів. Одним з недоліків цього ПЗ при порівнянні з поставленими цілями є відсутність можливості ручного створення та редагування зоряних систем, що є важливим елементом при виконанні користувацьких розрахунків та буде реалізовано у власному інтелектуальному помічнику.

The screenshot displays the 'Protodisc' application window. It features a 'Star System' section with fields for Name (Centra), Age (4.9), In Open Cluster (checked), and Population (Intermediate Population I). A 'Generate' button is present. Below this is a 'Database' section with Save, Load, and Delete buttons. The 'Primary Star' section includes fields for Name (Centra A), Spectral Type (M4), Temperature (3300), Mass (0.3), Luminosity Class (V), Luminosity (0.024), Radius (0.0022), and Snow Line (0.75). A table below lists 'Orbits: 10'. The 'Companion Star' section includes fields for Name (Centra B), Temperature (3100), Mass (0.1), Luminosity (0.0012), Spectral Type (M7), Radius (0.0006), Luminosity Class (V), Snow Line (0.17), Separation (Wide), Orbital Radius (70), Eccentricity (0.7), and Orbital Period (926.01). A table below lists 'Orbits: 8'. The 'Subcompanion Star' section includes fields for Name (Centra C), Temperature (3100), Mass (0.1), Luminosity (0.0012), Spectral Type (M7), Radius (0.0006), Luminosity Class (V), Snow Line (0.17), Separation (Distant), Orbital Radius (350), Eccentricity (0.5), and Orbital Period (10353.14). A table below lists 'Orbits: 6'. The 'Companion Star' section includes fields for Name (Centra D), Temperature (3100), Mass (0.1), Luminosity (0.0012), Spectral Type (M7), Radius (0.0006), Luminosity Class (V), Snow Line (0.17), Separation (Very Close), Orbital Radius (0.35), Eccentricity (0.4), and Orbital Period (0.46). A table below lists 'Orbits: 6'. A 'Close' button is at the bottom right.

Рисунок 1.7 – Програма “Protodisc”

1.3 Основні проблеми інтелектуалізації настільно-рольових ігор

Під час проведення сесії настільно-рольової гри важливим є час, який гравцями витрачається на саму гру. Одним з процесів який займає час гравців є керування ресурсами, доступними як групі, так і окремим гравцям. Цей процес не обмежується відніманням витрат від запасів, іноді включаючи розрахунок витрат, якщо такі не є очевидними.

Прикладом такої ситуації в обраній предметній області є розрахунок часу, необхідного на виконання маневру космічним кораблем, та палива, яке витрачається на такий маневр. Якщо такий маневр відомий заздалегідь, гравці чи майстер гри може виконати розрахунки до початку гри та використовувати результати при виникненні ситуації, яка цього вимагає.

Недоліком такого підходу є необхідність визначення маневру наперед в грі, яка підтримує повну свободу дій гравців та можливість настання випадкових подій внаслідок взаємодії гравців, ігрового світу, та результату кидка гральних кубиків.

У випадку коли ігрова ситуація змінюється від передбаченої, розрахунки необхідно виконувати ще раз. Додатково, якщо в грі ресурси гравців обмежені, що вірогідно при використанні таких правил, очікуваною поведінкою гравців є планування подальших дій, що включає порівняння доступних варіантів – навіть якщо гравці будуть виконувати попередньо визначений маневр, їм може бути цікаво скільки палива та часу знадобиться для виконання інших.

Якщо гравці планують виконання декількох маневрів, кількість рахунків відповідно збільшується, займаючи все більше або ігрового часу, або вимагаючи додаткового часу при підготовці до гри.

Можливим рішенням є планування всіх можливих маневрів наперед та використання такої таблиці під час гри, але для зоряної системи з дев'ятьма планетами, така таблиця вимагатиме виконання семидесяти двох розрахунків для кожного варіанту маневру. При використанні трьох можливих трансферів, описаних в статті “Halfway to Anywhere” [25], кількість таких підрахунків

дорівнює двісті шістнадцяти, а для перельоту, описаному в GURPS Spaceships [18] формулу наперед прорахувати неможливо, через можливість використання різної кількості палива для маневру.

Гравці можуть спростити процес розрахунків використовуючи ПЗ яке підтримує використання змінних при розрахунках, як Excel чи онлайн-калькулятори, але такі рішення можуть бути незадовільними, якщо користувачам необхідно кожного разу вводити дані для розрахунків.

При самотійному виконанні підрахунків виникає ще дві проблеми:

- Необхідність розуміння формул: В залежності від того як часто гравцями використовуються ці формули, може виникати необхідність повторного ознайомлення з ними та уточнення одиниць виміру, які використовуються в них. Для гравців, які використовують їх вперше, складність формул може відштовхнути їх від використання цих формул в ігровому процесі.

- Можливість допущення помилки гравцями при виконанні розрахунків. Виконання розрахунків, описаних в [25], вимагає використання декількох формул для визначення витрачених часу та палива, тож помилка в одній частині, призведе до некоректних результатів в майбутніх формулах, вимагаючи або виконання всіх підрахунків заново, або аналізу результатів для пошуку помилки.

Використання ПЗ, наприклад інтелектуального помічника, для виконання цих підрахунків дозволить спростити цей процес. Інтелектуальний помічник може виконувати розрахунки набагато швидше за користувача, не допускає помилок при розрахунках, та не втомлюється від їхнього виконання. Якщо в даних користувача наявна помилка, інтелектуальний помічник може виконувати перевірку та повідомляти його про це, уникаючи ситуацій, коли розрахунки виконуються для некоректних даних. На відміну існуючого рішення у вигляді Excel-файлу [25], наявність можливості зберігання та використання інформації про космічні кораблі та зоряні системи у

інтелектуального помічника дозволить спростити порівняння різних варіантів орбітальних маневрів під час ігрового процесу.

Важливою частиною ігор, які передбачають подорож в космосі є космічний корабель за допомогою якого пересуваються гравці. GURPS Spaceships [18] надає гравцям систему для опису такого космічного корабля, спрощуючи цей процес до вибору систем для кожного з двадцяти доступних слотів, визначення додаткових опцій, та виконання простих розрахунків похідних характеристик.

Цей підхід також може бути покращений з використанням інтелектуального помічника, автоматизуючи розрахунки на основі обраних користувачем систем та опцій, і також слідкуючи за правильністю побудови космічного корабля, попереджаючи користувача про помилки. Можливість вибору систем космічного корабля та миттєвого виконання розрахунків спростить цей процес в порівнянні з записуванням систем та їхніх характеристик вручну.

Враховуючи наявні аналоги цієї частини функціоналу інтелектуального помічника, її реалізація буде корисною не лише в межах пришвидшення розрахунків для орбітальних маневрів, але й як окрема від цього частина, забезпечуючи користувачам функціонуючий метод створення космічних кораблів за допомогою ПЗ.

2 АНАЛІЗ ШЛЯХІВ СТВОРЕННЯ ІНТЕЛЕКТУАЛЬНОГО ПОМІЧНИКА

2.1 Алгоритми розрахунку параметрів суб'єктів інтелектуального помічника

Суб'єктами інтелектуального помічника, зокрема, є орбітальні маневри та характеристики космічного корабля.

Перед розглядом формул, які стосуються запасу палива космічного корабля, також відомого як реакційна маса, необхідно зрозуміти як система опису космічних кораблів GURPS Spaceships [18] визначає поняття запасу палива їх.

В межах цієї системи опису космічних кораблів, яка буде реалізована в інтелектуальному помічнику, запас палива космічного корабля позначається як ΔV та вимірюється в милях на секунду. Для космічного корабля це відповідає можливій зміні швидкості відносно поточної, відносно певної точки в просторі. Наприклад, якщо космічний корабель рухається зі швидкістю 10 миль на секунду відносно цієї точки та в її напрямку, і має запас палива 20 миль на секунду, він може:

- прискоритись не більше ніж на 20 миль на секунду в напрямку точки, досягаючи швидкості 30 миль на секунду відносно точки, витративши весь запас палива;
- сповільнитись шляхом прискорення на 10 миль на секунду в протилежному напрямку від точки, фактично зупинившись відносно певної точки, залишаючи запас палива на подальшу зміну швидкості на 10 миль на секунду
- прискоритись до 20 миль на секунду в протилежному напрямку від точки, та досягти швидкості 10 миль на секунду в протилежному від точки напрямку, витративши весь запас палива;

- прискоритись в іншому напрямку, все ще рухаючись “вперед” зі швидкістю 10 миль на секунду, але також зміщуючись в сторону відносно певної точки.

Окрім запасу палива космічного корабля, за допомогою ΔV може бути позначена кількість палива, необхідна для виконання маневру – вартість виконання пришвидшення на 10 миль на секунду, це кількість палива, яка відповідає 10 милям на секунду.

В милях на секунду також вимірюється швидкість, необхідна для покидання поверхні планети. Така швидкість називається другою космічною, і відповідає необхідній кількості палива, необхідного для виконання такої дії.

Розрахунок необхідного палива та витраченого часу для виконання орбітальних маневрів за формулами, описаними в [25] та [18], вимагає їхнього аналізу для коректної реалізації в інтелектуальному помічнику.

Гоманівська траекторія вимагає двох прискорень. Кількість палива, використаного на перше прискорення вираховується за наступною формулою:

$$\Delta V_1 = 18.52 * \sqrt{\frac{M}{R_i}} * \left(\sqrt{\frac{2 * R_f}{R_i + R_f}} - 1 \right) \quad (1)$$

де: ΔV_1 – паливо, витрачене на перше прискорення.

M – маса зірки в масах Сонця.

R_i – радіус початкової орбіти в астрономічних одиницях.

R_f – радіус фінальної орбіти в астрономічних одиницях.

Кількість палива, використаного для другого прискорення отримується за формулою:

$$\Delta V_2 = 18.52 * \sqrt{\frac{M}{R_f}} * \left(1 - \sqrt{\frac{2 * R_i}{R_i + R_f}} \right) \quad (2)$$

де: ΔV_2 – паливо, витрачене на друге прискорення.

M – маса зірки в масах Сонця.

R_i – радіус початкової орбіти в астрономічних одиницях.

R_f – радіус фінальної орбіти в астрономічних одиницях.

Модуль суми результатів цих формул (ΔV_1 та ΔV_2) дорівнює загальній кількості витраченого палива на виконання маневру. Сума може бути негативною – в такому випадку прискорення відбувається проти напрямку поточної орбіти.

Час, витрачений на виконання маневру можна визначити за формулою:

$$T = 3.14 * C * \sqrt{\frac{(R_i + R_f)^3}{8 * M}} \quad (3)$$

де: T – час, витрачений на маневр.

C – 58.11 для результату в днях, 0.159 для результату в роках.

M – маса зірки в масах Сонця.

R_i – радіус початкової орбіти в астрономічних одиницях.

R_f – радіус фінальної орбіти в астрономічних одиницях.

Для біеліптичної траєкторії необхідно виконати три прискорення, відповідно витрачене паливо розраховується по трьом формулам. Для цих формул також необхідно знати розміри половини більших осей еліптичних трансферних орбіт.

Перша формула, яка дозволяє дізнатись кількість палива, використаного для першого прискорення:

$$\Delta V_1 = 18.52 * \left(\sqrt{\frac{2 * M}{R_i} - \frac{M}{a_1}} - \sqrt{\frac{M}{R_i}} \right) \quad (4)$$

де: ΔV_1 – паливо, витрачене на перше прискорення.

a_1 – половина більшої осі першої еліптичної орбіти.

M – маса зірки в масах Сонця.

R_i – радіус початкової орбіти в астрономічних одиницях.

Для знаходження кількості палива, використаного для другого прискорення, використовується формула:

$$\Delta V_2 = 18.52 * \left(\sqrt{\frac{2 * M}{R_c} - \frac{M}{a_2}} - \sqrt{\frac{2 * M}{R_i} - \frac{M}{a_1}} \right) \quad (5)$$

де: ΔV_2 – паливо, витрачене на друге прискорення.

a_1 – половина більшої осі першої еліптичної орбіти.

a_2 – половина більшої осі другої еліптичної орбіти.

M – маса зірки в масах Сонця.

R_i – радіус початкової орбіти в астрономічних одиницях.

R_c – відстань від зірки до пікової точки першого прискорення в астрономічних одиницях.

R_f – радіус фінальної орбіти в астрономічних одиницях.

Для третього прискорення, кількість витраченого палива знаходиться за формулою:

$$\Delta V_3 = 18.52 * \left(\sqrt{\frac{2*M}{R_f} - \frac{M}{a_2}} - \sqrt{\frac{M}{R_f}} \right) \quad (6)$$

де: ΔV_3 – паливо, витрачене на третє прискорення.

a_2 – половина більшої осі другої еліптичної орбіти.

M – маса зірки в масах Сонця.

R_f – радіус фінальної орбіти в астрономічних одиницях.

Знайшовши суму модулів ΔV_1 , ΔV_2 та ΔV_3 , можна отримати загальні витрати пального на подорож. Використання модулів важливе, так як хоча б одне зі значень буде від'ємне через необхідність зменшення швидкості космічного корабля.

Для знаходження часу між першим та другим прискореннями необхідно скористатись наступною формулою:

$$T_1 = 3.14 * C * \sqrt{\frac{a_1^3}{M}} \quad (7)$$

де: T_1 – час між першим та другим прискореннями.

C – 58.11 для результату в днях, 0.159 для результату в роках.

a_1 – половина більшої осі першої еліптичної орбіти.

M – маса зірки в масах Сонця.

Для знаходження часу між другим та третім прискореннями необхідно скористатись наступною формулою:

$$T_2 = 3.14 * C * \sqrt{\frac{a_2^3}{M}} \quad (7)$$

де: T_2 – час між другим та першим прискореннями.

C – 58.11 для результату в днях, 0.159 для результату в роках.

a_2 – половина більшої осі другої еліптичної орбіти.

M – маса зірки в масах Сонця.

Брахістохронна траекторія вимагає постійного прискорення, спочатку в напрямку обраної орбіти, а потім в зворотному, для зупинки на необхідній траекторії. Для визначення витрат палива використовується одна формула:

$$\Delta V = 1500 * \sqrt{A * (R_f - R_i)} + 18.52 * \sqrt{\frac{M}{R_f - R_i}} \quad (8)$$

де: ΔV – паливо, витрачене на третє прискорення.

A – прискорення космічного корабля в G .

M – маса зірки в масах Сонця.

R_i – радіус початкової орбіти в астрономічних одиницях.

R_f – радіус фінальної орбіти в астрономічних одиницях

Тривалість цього маневру вираховується за формулою:

$$T = J * \sqrt{\frac{R_f - R_i}{A}} + 9.8 * A * C * \sqrt{\frac{R_f - R_i}{M}} \quad (9)$$

де: T – час, витрачений на маневр.

C – 58.11 для результату в днях, 0.159 для результату в роках.

J – 2.86 для результату в днях, 0.00783 для результату в роках.

A – прискорення космічного корабля в G .

M – маса зірки в масах Сонця.

R_i – радіус початкової орбіти в астрономічних одиницях.

R_f – радіус фінальної орбіти в астрономічних одиницях.

Для визначення другої космічної швидкості, яка необхідна для зльоту з планети та покидання її орбіти, було використано формулу, запозичену в [25]. Результат цієї формули відповідає запасу палива, необхідного для виконання такого маневру. Формула має наступний вигляд:

$$V_{esc} = 6.95 * \sqrt{M/R} \quad (10)$$

де: V_{esc} – друга космічна швидкість та необхідна кількість палива;

M – маса планети виражена в масах Землі;

R – радіус планети, виражений в радіусах Землі.

Якщо космічний корабель має крила чи систему антигравітації та реактивні двигуни чи двигуни які можуть працювати в атмосфері без використання палива, при зльоті з планети застосовується наступна формула для знаходження зекономленої кількості палива:

$$\Delta V_s = S/3600 \quad (11)$$

де: ΔV_s – зекономлена кількість пального в милях на секунду;

S – атмосферна швидкість космічного корабля в милях на годину з використанням лише реактивних двигунів, чи двигунів які працюють в атмосфері без використання палива.

Для визначення запасу палива, необхідного лише на покидання орбіти планети, було використано наступну формулу [25]:

$$\Delta V = 2.03 * \sqrt{M/R} \quad (12)$$

де: ΔV – необхідний запас палива для покидання орбіти;

M – маса планети виражена в масах Землі;

R – радіус планети, виражений в радіусах Землі.

Для визначення часу, витраченого на зліт та покидання орбіти, чи лише на покидання орбіти, використовується наступна формула [25]:

$$T = 4.39 * (\Delta V/A) \quad (13)$$

де: T – витрачений на маневр час, в годинах;

ΔV – необхідна для маневру кількість палива, в милях на секунду;

A – прискорення космічного корабля в G.

Так як деякі орбіти деяких планет можуть знаходитись в різних площинах, може виникнути необхідність здійснити додатковий маневр для переходу на обрану орбіту. Розрахунок необхідного палива та часу реалізовується за наступною формулою [25]:

$$\Delta V = 2 * B * \sqrt{\frac{M}{R}} * \sin\left(\frac{\theta}{2}\right) \quad (14)$$

де: ΔV – необхідний запас палива для виконання маневру;

B – константа 18.52;

R – відстань до зірки, виражена в астрономічних одиницях;

Θ – кут нахилу однієї орбітальної площини відносно іншої.

Для визначення витраченого на цей маневр часу, використовується наступна формула [25]:

$$T = (2 * R * \sin(\Theta)) / (K * \Delta V) \quad (15)$$

де: T – витрачений на маневр час, в годинах чи роках;

R – відстань до зірки, виражена в астрономічних одиницях;

K – дорівнює 0.000932 для отримання часу в днях чи 0.33 для отримання часу в роках;

ΔV – витрачена на маневр кількість палива, в милях на секунду;

Θ – кут нахилу однієї орбітальної площини відносно іншої.

Розрахунок характеристик космічного корабля є важливою частиною функціоналу не тільки при створенні космічного корабля, але й при виконанні розрахунків орбітальних маневрів, що призводить до необхідності аналізу цих розрахунків перед реалізацією в інтелектуальному помічнику.

Атмосферна швидкість космічного корабля вимірюється в милях на годину. Ця характеристика може бути корисною при спробі злетіти з планети з атмосферою за наявності у космічного корабля крил чи антигравітаційної системи, або ж при використанні космічного корабля в атмосфері впродовж більшої кількості часу ніж необхідно на зліт з планети. Атмосферна швидкість вираховується за наступною формулою:

$$S = \sqrt{A/D} \quad (16)$$

де: S – атмосферна швидкість в милях на годину.

A – прискорення в G .

D – дорівнює 1 якщо космічний корабель аеродинамічний, та 10 якщо ні.

Кількість екіпажу космічного корабля розраховується шляхом знаходження суми робочих місць для всіх систем космічного корабля, множення цієї кількості на коефіцієнт автоматизації, який залежить від обраного рівня автоматизації, та додаванням кількості місць на всіх містках космічного корабля. Кількість місць в каютах може бути меншою за кількість екіпажу. В такому випадку корабель не є придатним для довготривалого

використання, так як запаси їжі та повітря, згідно правил, обмежені 24 годинами.

Вартість космічного корабля розраховується шляхом знаходження суми фінальних вартостей систем космічного корабля, множенням початкової кількості робочих місць на вартість автоматизації робочого місця, та додаванням вартості таких опцій як крила, обертальна гравітація, маскування, динамічне візуальне маскування, якщо такі наявні.

Максимальне використання енергії розраховується знаходженням суми використання енергії всіма системами космічного корабля. Виробництво енергії дорівнює сумі виробництва енергії всіх силових установок космічного корабля. Важливо зауважити що перевищення максимальним використанням виробництва є допустимою ситуацією, так як не всі системи які потребують енергії повинні працювати одночасно.

Рівень здоров'я космічного корабля розраховується з базового числа 13, до якого додається 1 при відсутності автоматизації, наявності хоча б однієї фабрики, але віднімається 1 при застосуванні високого рівня чи повної автоматизації при восьмому чи нижчому технологічному рівні, при відсутності інженерної кімнати для космічного корабля з модифікатором розміру меншим за 9, та віднімається 2 чи 4 при використанні поганої чи дуже поганої якості космічного корабля.

Рівень керованості відповідає обраному розміру космічного корабля та додатково модифікується в залежності від прискорення космічного корабля. Рівень стабільності відповідає обраному розміру космічного корабля.

Для знаходження атмосферних рівнів цих характеристик, до рівня керованості додається 2, якщо космічний корабель має антигравітаційну систему та 4 за наявності крил. Якщо одночасно застосовуються обидва бонуси, додається 5 замість 6. Максимальне значення керованості дорівнює 5. За наявності крил, атмосферний рівень стабільності збільшується на 1.

Рівні броні для кожної секції космічного корабля знаходяться шляхом додавання значень броні кожної системи броні в секції, а рівень захисту

силових полів знаходиться шляхом додавання рівнів броні всіх систем силових полів. Додатково, за правилами, наведеними в [28], якщо на космічному кораблі є більше п'яти систем броні, рівень її захисту додатково збільшується.

Так як одна система відображає $1/20$ маси космічного корабля та броня є важкою відносно її об'єму, внаслідок чого займає менше об'єму, наявність декількох систем броні достатньо зменшує площу поверхні космічного корабля для покриття товстішим шаром броні при збереженні однакової маси.

Додатково, при використанні великої кількості броні, розмір космічного корабля зменшується достатньо для зменшення його ефективного модифікатора розміру. Цей зменшений модифікатор не впливає на вартість систем космічного корабля, але впливає на складність влучання чи детекції космічного корабля.

Коефіцієнт збільшення захисного ефекту броні та впливу кількості броні на ефективний модифікатор розміру зображені в табл. 2.1. Таблиця закінчується на 18 системах броні, так як дві з систем космічного корабля мають тип "Core", тобто розміщені в центрі космічного корабля, і не можуть містити систему броні.

Таблиця 2.1 – Вплив систем броні на рівень захисту та розмір

Кількість систем броні	Коефіцієнт	Модифікатор розміру
1	2	3
0-4	1	0
5-6	1.2	0
7	1.3	0
8-9	1.4	0
10	1.5	0
11	1.6	0

Таблиця 2.1 – Продовження

1	2	3
12-13	1.8	-1
14	2	-1
15	2.3	-1
16	2.6	-1
17	3	-1
18	3.6	-2

Максимальне прискорення дорівнює сумі прискорень всіх двигунів, для яких космічний корабель має відповідне паливо.

Запас палива космічного корабля дорівнює сумі паливних баків з паливом одного типу, помноженій на значення ефективності використання палива найбільш ефективного двигуна для цього типу палива та коефіцієнт, який залежить від кількості баків (табл. 2.2). Множення на цей коефіцієнт відбувається через зменшення маси космічного корабля, пов'язаного з використанням певної кількості палива під час маневрів, що дозволяє використовувати менше палива для подальшого прискорення. За наявності двигуна, який не потребує палива, запас палива дорівнює нескінченності.

Таблиця 2.2 – Коефіцієнт доступного палива

Кількість баків	Коефіцієнт
1	2
0-5	1
6-8	1.2
9-12	1.4
13-14	1.6
15	1.8

Таблиця 2.2 – Продовження

1	2
16	2
17	2.2
18	2.5
19	3

Рівень потужності комп'ютерних систем [20] залежить від наявності містка та технологічного рівня космічного корабля. Він знаходиться шляхом додавання модифікатора потужності комп'ютерних систем для обраного технологічного рівня до базового рівня потужності комп'ютерних систем містка. За відсутності містка, комп'ютерні системи відсутні та їхній рівень дорівнює нулю.

Рівень сенсорної системи залежить від наявності містка чи окремої сенсорної системи і технологічного рівня космічного корабля. До технологічного рівня космічного корабля додається модифікатор сенсорної системи. Якщо наявна окрема сенсорна система, використовується її модифікатор. За її відсутності, використовується модифікатор сенсорної системи містка. Якщо він також відсутній, космічний корабель не має сенсорної системи.

2.2 Архітектура інтелектуального помічника “Captain’s Companion”

Для інтелектуального помічника було обрано клієнт-серверну архітектуру. Така архітектура характеризується наявністю клієнтського застосунку, з яким взаємодіє користувач, та сервера, з яким взаємодіє клієнтський застосунок шляхом надсилання запитів. В даному випадку в ролі

клієнтського застосунку виступає розроблювана програма, а в ролі сервера – Microsoft SQL Server, який забезпечує роботу бази даних.

Таке рішення було прийняте для спрощення роботи інтелектуального помічника зі збереженими даними.

Клієнтську частину інтелектуального помічника можна поділити на чотири блоки:

- стартовий блок;
- блок зоряних систем;
- блок космічних кораблів;
- блок орбітальних маневрів.

Стартовий блок не взаємодіє з базою даних, є першою частиною графічного інтерфейсу користувача з якою взаємодіє користувач при запуску інтелектуального помічника, та забезпечує перехід до інших блоків чи завершення роботи з інтелектуальним помічником.

Блок зоряних систем відповідає за взаємодію користувача з зоряними системами, дозволяє створення нових записів, редагування чи видалення існуючих, і також їхній перегляд. Цей блок отримує та змінює інформацію в базі даних. В цьому блоці відсутні переходи до інших блоків, окрім стартового.

Блок космічних кораблів відповідає за взаємодію користувача з космічними кораблями, дозволяє створення нових записів, редагування, видалення та перегляд існуючих. Цей блок отримує та змінює інформацію в базі даних. В цьому блоці відсутні переходи до інших блоків, окрім стартового.

Блок орбітальних маневрів відповідає за розрахунок орбітальних маневрів з використанням даних, введених користувачем, чи отриманих з бази даних. Цей блок лише отримує інформацію з бази даних та не може вносити до неї зміни. В цьому блоці відсутні переходи до інших блоків, окрім стартового.

Схематичний зв'язок блоків між собою та з базою даних описаний в блок-схемі, зображений на рис 2.1.

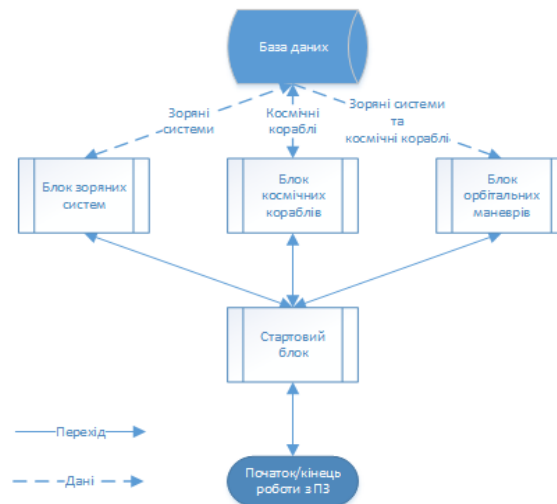


Рисунок 2.1 – Взаємодія блоків між собою та з БД

Стартовий блок інтелектуального помічника містить одну форму, яка є першою формою до якої користувач має доступ при початку роботи з інтелектуальним помічником, дозволяє користувачу перейти до інших блоків, та завершити роботу з інтелектуальним помічником. Її зв'язок з іншими блоками зображено на рис. 2.2

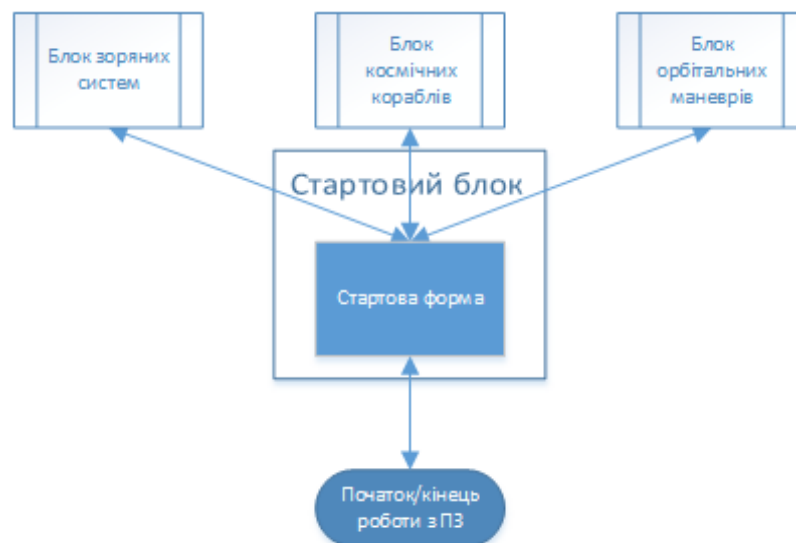


Рисунок 2.2 – Зв'язки стартового блоку

Блок зоряних систем інтелектуального помічника містить три форми: центральна форма, форма додавання/редагування, та форма перегляду.

Центральна форма є першою доступною формою користувачу при відкритті блоку. За допомогою неї виконується перехід до форми додавання/редагування та форми перегляду, і також до стартового блоку. Центральна форма отримує дані про зоряні системи та забезпечує їхній перегляд в текстовому вигляді та надає користувачу можливість видалити дані про зоряну систему з бази даних.

Форма додавання/редагування доступна користувачу з центральної форми та забезпечує створення нових записів в базі даних чи редагуванні існуючих, в залежності від обраної користувачем дії. Форма отримує дані з бази даних в межах підготовки до створення чи редагування запису, дані про існуючі зоряну систему при редагуванні отримуються з центральної форми.

Дані передаються в базу даних при створенні нового запису чи редагуванні існуючого. З цієї форми можливий перехід лише до центральної форми. Форма перегляду дозволяє користувачу переглянути графічне відображення існуючої системи у вигляді. З цієї форми не здійснюється обмін даними з базою даних. Дані про зоряну систему отримуються з центральної форми. Єдиною доступною формою для переходу є центральна форма.

Зв'язок форм всередині цього блоку та цього блоку з іншими та базою даних зображено на блок-схемі на рис. 2.3.

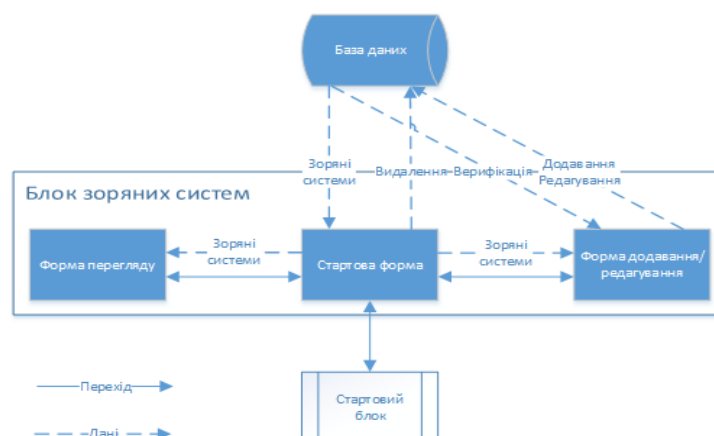


Рисунок 2.3 – Зв'язки блоку зоряних систем

Блок космічних кораблів інтелектуального помічника містить дві форми: головна форма та форма додавання/редагування.

Головна форма є першою доступною користувачу формою при переході до блоку зі стартового. Ця форма забезпечує перегляд інформації про космічні кораблі та можливість видалення записів про них. Форма отримує інформацію з бази даних та видаляє її. З цієї форми користувач може перейти до форми додавання/редагування чи до стартового блоку.

Форма додавання/редагування космічних кораблів дозволяє користувачу ввести дані про новий космічний корабель та зберегти їх, або ж внести та зберегти зміни в інформації про існуючий космічний корабель. Ця форма передає дані про космічні кораблі в базу даних. Інформацію про космічний корабель при редагуванні форма додавання/редагування отримує з головної форми. Форма отримує загальні дані про космічні кораблі з бази даних. З цієї форми користувач може перейти до головної форми.

Блок-схема зв'язків форм в цьому блоці та з базою даних чи іншими блоками зображена на рис. 2.4.

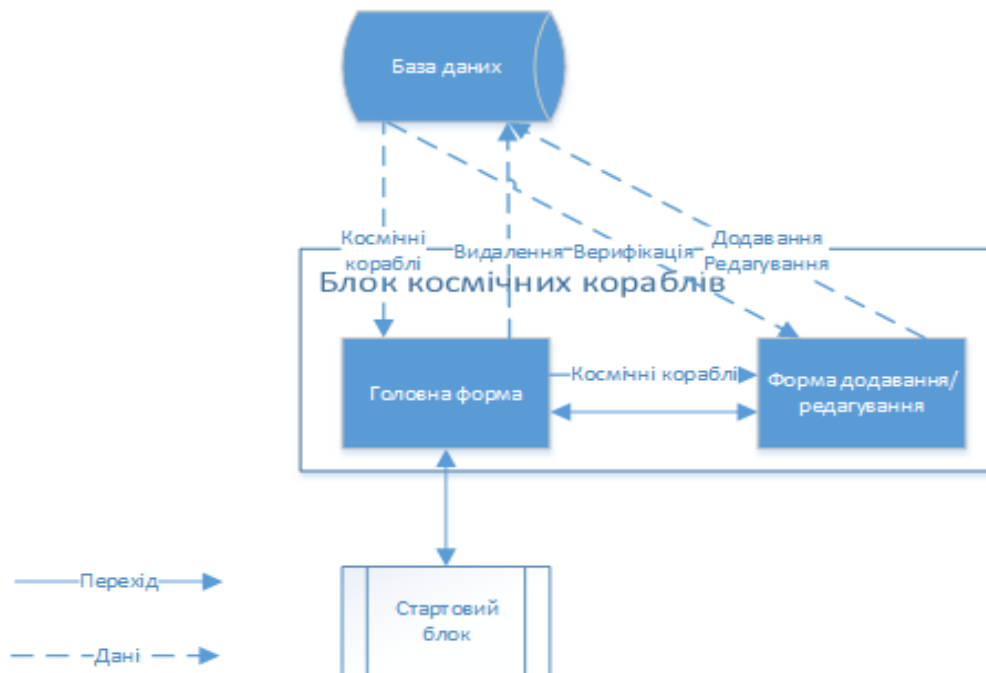


Рисунок 2.4 – Зв'язки блоку космічних кораблів

Блок орбітальних маневрів інтелектуального помічника містить одну форму – форму розрахунків. Форма розрахунків доступна користувачу після переходу до блоку орбітальних маневрів.

Ця форма дозволяє користувачу виконувати розрахунки орбітальних маневрів використовуючи власноруч введені дані чи дані, отримані з бази даних.

Ця форма отримує з бази даних інформацію про зоряні системи та космічні кораблі. Форма не має можливості внесення змін в базу даних. З цієї форми користувач може повернутись до стартового блоку. Блок-схема взаємодії блоку з іншими блоками та БД зображена на рис. 2.5.

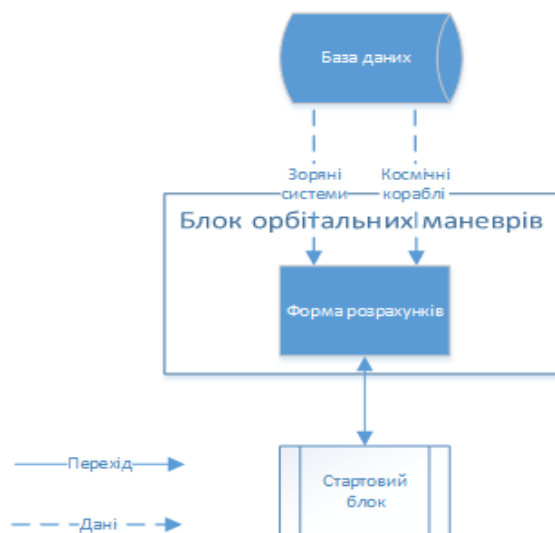


Рисунок 2.5 – Зв'язки блоку орбітальних розрахунків

Загальна блок-схема, на якій детально зображені всі блоки інтелектуального помічника, зображена на рис.

На цій блок-схемі зображені всі зв'язки всередині блоків, між блоками, та з базою даних.

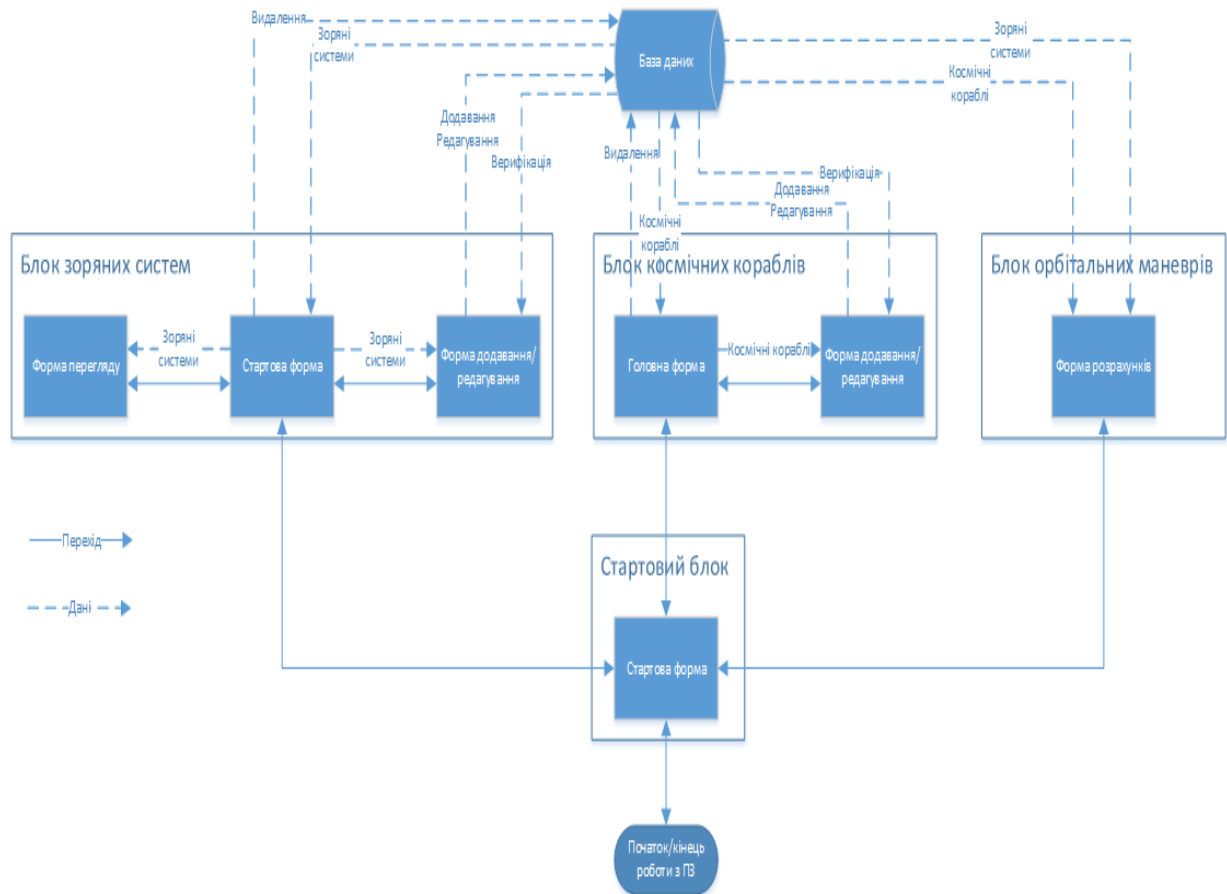


Рисунок 2.6 – Зв'язки всіх блоків

2.3 Опис використаних технологій розробки інтелектуального помічника

Клієнтська частина інтелектуального помічника була створена за допомогою мови програмування C#. Завдяки тому, що ця мова програмування підтримує об'єктно-орієнтоване програмування, процес написання програмного коду вдалось спростити завдяки використанню класів для зберігання даних та взаємодії з наборами різноманітних даних як з об'єктами.

Використання методів класів дозволило спростити отримання інформації про екземпляри класів, дозволяючи виконувати складні за допомогою мінімальної кількості додаткового коду та забезпечуючи

використання коректних даних для розрахунків, використовуючи лише ті дані, які вже належать екземпляру класу.

Важливим елементом клієнтської частини інтелектуального помічника є візуальний інтерфейс, який було реалізовано завдяки фреймворку графічного інтерфейсу користувача Windows Forms [34]. Цей фреймворк надає велику кількість різноманітних елементів керування, які дозволяють швидко створювати графічні інтерфейси користувача, без необхідності створення елементів керування власноруч.

Елементи керування, наявні у Windows Forms, дозволяють налаштовувати їх під потреби розробника, змінюючи їхній розмір, колір, текстові написи, фонові зображення, та інші характеристики, що дозволяє створити максимально зручний інтерфейс для майбутнього користувача, як візуально, так і ергономічно.

Окрім можливості налаштування, елементи керування Windows Forms підтримують обробку подій. Подією є певна дія користувача чи програми, яка впливає на елемент керування, внаслідок чого викликається пов'язаний з цією подією метод, виконуючи певні дії. Використання подій дозволяє створювати інтерфейси, які реагують на будь-які дії користувача, а не лише на базову його взаємодію з елементами керування.

Таким чином, можна створити програму, яка після введення тексту відразу починає його обробку, не вимагаючи додаткового натискання окремої кнопки користувачем. Так як події відбуваються і при внесенні змін програмою, з'являється можливість каскадного виклику подій, що може бути корисним в певних випадках, але також при неувважному використанні може призвести до замкнутого циклу, під час якого перша подія викликає другу, друга третю, а третя першу.

Розробка клієнтської частини інтелектуального помічника виконувалась в середовищі розробки Microsoft Visual Studio [35]. Це середовище розробки дозволяє виконувати написання програмного коду та створювати візуальні

інтерфейси, що було важливим під час створення інтелектуального помічника, та виконувати інші важливі завдання при створенні ПЗ.

Однією з корисних функцій, які забезпечує Visual Studio є перевірка програмного коду на коректність написання в реальному часі. Таким чином помилки, які можуть виникати під час написання коду, підсвічуються після їхнього допущення, даючи знати користувачеві про їхню наявність. Пов'язані з помилками повідомлення допомагають у їхньому вирішенні, повідомляючи користувача про суть помилки.

Під час запуску ПЗ за допомогою Visual Studio, при виникненні помилок в ході роботи ПЗ, середовище розробки також вказує на рядок, в якому виникла помилка з інформацією щодо її характеру.

Під час написання програмного коду, Microsoft Visual Studio також пропонує автоматичне доповнення введених команд чи назв змінних на основі вже існуючих в файлі з кодом та найчастіше використовуваних під час написання коду.

Можливість взаємодії з графічним інтерфейсом користувача створюваного ПЗ за допомогою графічного інтерфейсу середовища розробки спрощує його створення та дозволяє робити макети, які можна переглядати до запуску ПЗ чи процесу відладки.

Так як ця можливість підтримує роботу з Windows Forms, процес створення графічного інтерфейсу користувача для інтелектуального помічника було значно спрощено.

Серверна частина клієнт-серверної архітектури інтелектуального помічника була реалізована за допомогою Microsoft SQL Server [36], системи керування реляційними базами даних. Ця система керування базами даних дозволяє виконувати стандартні операції над базами даних, такі як зчитування, запис, модифікація, та видалення даних, і також створювати та викликати збережені процедури, тож вона задовольняє вимогам завдань, які ставляться перед інтелектуальним помічником при роботі зі збереженими даними. Окрім цього, ця система підтримує виконання резервних копій, дозволяючи

користувачу мігрувати створені в ході роботи з інтелектуальним помічником дані на інший пристрій та продовжити роботу з нього, або ж убезпечитись від втрати даних у разі виникнення збою в роботі обладнання чи пошкодження даних.

Створення бази даних було виконано використовуючи середовище SQL Server Management Studio для системи керування базами даних Microsoft SQL Server 2008. Це середовище підтримує написання та виконання команд мовою запитів T-SQL (Transact-SQL).

Під час написання скрипту для створення бази даних, середовище пропонує назви таблиць та полів які було прописано але ще не було створено, спрощуючи написання великих скриптів. Можливість відокремлення частин скрипта для послідовного виконання дозволяє виконувати великі скрипти, частини яких покладаються на попереднє виконання певних дій, наприклад внесення даних в таблиці, що створюються.

Для реалізації зв'язку між клієнтською та серверною частиною інтелектуального помічника, було використано бібліотеку Microsoft.Data.SqlClient [37]. Ця бібліотека є забезпечує роботу з реляційними базами даних, такими як Microsoft SQL Server. За допомогою неї можна виконувати підключення як до локальних баз даних, так і до віддалених, використовуючи різні типи авторизації.

При виконанні запитів, відповідні класи та методи підтримують використання параметрів, які можуть використовуватись для динамічної зміни запитів при виклику їх в програмному коді, підставляючи необхідні значення в текст запиту, без необхідності створення повністю нового тексту. Цю бібліотеку було створено на основі System.Data.SqlClient, доповнюючи її функціонал та покращуючи безпеку обміну даних з базою даних.

3 ОПИС ПРОГРАМНОГО ПРОДУКТУ “CAPTAIN’S COMPANION”

3.1 Вимоги до інтелектуального помічника

Основними нефункціональними вимогами до програмного продукту є:

- наявність валідації даних – введені користувачем дані перевіряються на відповідність очікуванім, у випадку якщо дані не відповідають очікуванім, операція не виконується та користувача повідомляється про причину відмови;
- відмовостійкість – дії користувача або пошкодження даних в базі даних чи під час передачі не повинні призводити до аварійної зупинки ПЗ;
- швидкодія – виконання будь-якої дії займає не більше п’яти секунд;
- інтуїтивний інтерфейс – інтерфейс повинен бути простим в навігації та дозволяти користувачам, знайомим з системою опису космічних кораблів GURPS Spaceships [18], швидко та ефективно виконувати необхідні дії.

Функціональні вимоги до інтелектуального помічника розділено за трьома блоками: блок роботи з зоряними системами, блок роботи з космічними кораблями, та блок розрахунку орбітальних маневрів.

До блоку роботи з зоряними системами висуваються наступні функціональні вимоги:

- можливість додавання, редагування, та видалення даних про зоряні системи;
- можливість перегляду даних про зоряну систему в текстовому та графічному виглядах;
- заборона наявності двох зоряних систем чи небесних тіл з однаковою назвою;
- відсутність можливості введення значень маси, радіусу небесного тіла, та сили тяжіння для поясів астероїдів.

До блоку роботи з космічними кораблями висуваються наступні функціональні вимоги:

- можливість додавання, редагування та видалення інформації про космічні кораблі;
- можливість перегляду інформації про космічні кораблі в текстовому вигляді;
- заборона наявності двох космічних кораблів з однаковою назвою;
- створення космічних кораблів повинно включати вибір систем для 20 слотів космічного корабля;
- можливість додаткового налаштування обраних систем згідно їхнього типу;
- автоматичне вилучення систем космічного корабля зі слотів, якщо внаслідок внесення змін вони стають недоступними для вибору;
- можливість встановлення рівнів автоматизації та стану космічного корабля;
- можливість встановлення таких ознак як аеродинамічність, наявність крил, обертальної гравітації, маскуванню, та динамічного візуального маскуванню;
- виконання помічником підрахунків наступних характеристик:
 - вартість космічного корабля;
 - рівень надійності;
 - кількість екіпажу;
 - максимальне прискорення в G;
 - запас палива;
 - максимальна атмосферна швидкість;
 - рівні броні для секцій корпусу космічного корабля.

До блоку розрахунків орбітальних маневрів висуваються наступні функціональні вимоги:

- можливість введення даних для розрахунків користувачем чи вибору існуючих даних за допомогою випадających списків;

- можливість вибору між чотирма орбітальними маневрами – гоманівським трансфером, біеліптичним трансфером, брахістохронним трансфером, та перельотом між двома точками;
- можливість виконання розрахунків з урахуванням необхідності зльоту з планети, покидання орбіти, чи ігноруванням такої опції;
- розрахунок економії палива при можливості зльоту з планети за наявності атмосфери та відповідної конструкції космічного корабля;
- повідомлення користувача при неможливості виконання маневру з вказанням причини.

3.2 Структура бази даних програмного продукту

База даних, розроблена для зберігання даних, необхідних для функціонування інтелектуального помічника, та даних, згенерованих внаслідок взаємодії користувача з інтелектуальним помічником, складається з двох частин – частина зоряних систем та частина космічних кораблів.

Частина зоряних систем містить три таблиці: “star_systems” (зоряні системи), “stellar_object_types” (типи небесних тіл), та “stellar_objects” (небесні тіла).

Таблиця “star_systems” містить інформацію про зоряну систему та її зірку. Цією інформацією є назва зоряної системи, яка відповідає назві зірки, та маса зірки в масах Сонця (табл.3.1).

Таблиця 3.1 – Структура таблиці “star_systems”

Найменування	Ідентифікатор	Тип поля	Примітки
Код зоряної системи	id	int	PK, NN
Назва зоряної системи	name	varchar(20)	U, NN
Маса зірки	mass	float	NN

Таблиця “stellar_object_types” містить інформацію про типи небесних тіл, які знаходяться в зоряній системі. Цією інформацією є назва типу. Структура цієї таблиці зображена в табл. 3.2.

Таблиця 3.2 – Структура таблиці “stellar_object_types”

Найменування	Ідентифікатор	Тип поля	Примітки
Код типу небесного тіла	id	int	PK, NN
Назва типу небесного тіла	name	varchar(17)	U, NN

Таблиця “stellar_objects” містить інформацію про небесні тіла, які знаходяться в зоряній системі. Ця таблиця містить назву, орбітальний радіус в астрономічних одиницях, ексцентриситет орбіти, нахил площини орбіти в градусах, масу об’єкта в масах Землі, радіус об’єкта в радіусах Землі, силу тяжіння в G, та опис об’єкта. Додатково, в таблиці зберігається id-код зоряної системи, до якої належить небесне тіло, та id-код типу небесного тіла, яким є небесне тіло. Структура цієї таблиці зображена в табл. 3.3.

Таблиця 3.3 – Структура таблиці “stellar_objects”

Найменування	Ідентифікатор	Тип поля	Примітки
1	2	3	4
Код небесного тіла	id	int	PK, NN
Код зоряної системи	star_system	int	FK(stellar_objects.id), NN
Код типу небесного тіла	stellar_object_type	int	FK(stellar_object_types.id), NN
Назва небесного тіла	name	varchar(20)	NN, U
Орбітальний радіус	orbital_radius	float	NN

Таблиця 3.3 – Продовження

1	2	3	4
Ексцентриситет орбіти	orbital_eccentricity	float	NN
Нахил орбітальної площини	orbital_angle	int	NN
Маса небесного тіла	mass	float	NN
Радіус небесного тіла	planetary_radius	float	NN
Сила тяжіння	gravity	float	NN
Опис небесного тіла	object_description	varchar(500)	NN

На рис 3.1. зображена діаграма частини бази даних, яка відповідає за зоряні системи та небесні тіла.

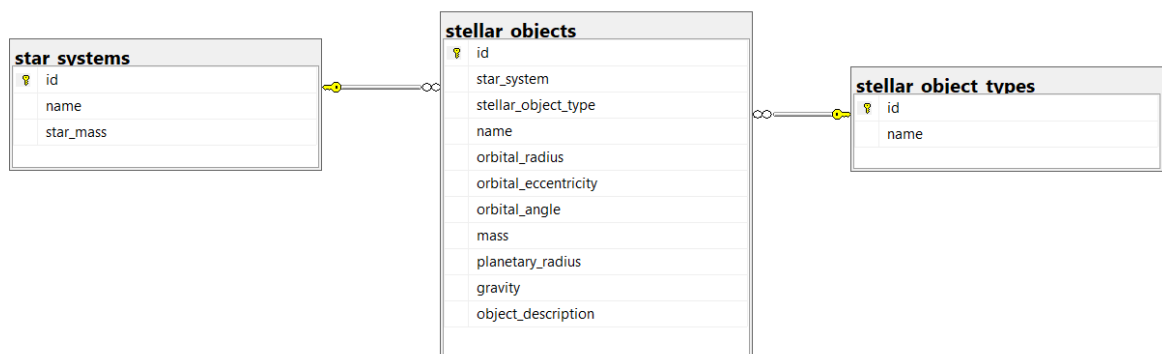


Рисунок 3.1 – Діаграма частини бази даних, яка стосується зоряних систем

Частина космічних кораблів містить 84 таблиці, які призначені для зберігання створених користувачем космічних кораблів та фіксованої інформації про доступні системи та характеристики.

Головною таблицею для цієї частини бази даних є “spaceships” (космічні кораблі). Ця таблиця призначена для зберігання інформації про космічні кораблі, створені користувачем.

Ця таблиця пов'язана з таблицями “tl” (технологічні рівні), “sm” (розмір), “automation” (автоматизація), та “condition” (стан) зовнішніми ключами. Структура цієї таблиці зображена в табл. 3.4.

Таблиця 3.4 – Структура таблиці “spaceships”

Найменування	Ідентифікатор	Тип поля	Примітки
Код космічного корабля	id	int	PK, NN
Назва космічного корабля	name	varchar(50)	NN, U
Технологічний рівень космічного корабля	tl	int	FK(tl.id), NN
Модифікатор розміру космічного корабля	sm	int	FK(sm.id), NN
Аеродинамічність	streamlined	bit	NN
Наявність крил	winged	bit	NN
Наявність обертальної гравітації	spin_gravity	bit	NN
Наявність маскуванню	stealth	bit	NN
Наявність динамічного візуального маскуванню	dynamic_chameleon	bit	NN
Рівень автоматизації	automation	int	FK(automation_levels.id), NN
Стан космічного корабля	condition	int	FK(ship_conditions.id), NN

Таблиця “tl” зберігає дані про технологічні рівні, від яких залежить доступність та характеристики систем космічного корабля. Її структура зображена в табл. 3.5.

Таблиця 3.5 – Структура таблиці “tl”

Найменування	Ідентифікатор	Тип поля	Примітки
Код технологічного рівня	id	int	PK, NN
Назва технологічного рівня	name	varchar(21)	U, NN
Модифікатор обчислювальної потужності	computer_comple xity		NN

Таблиця “sm” зберігає дані про модифікатори розміру космічного корабля, від яких залежить довжина та маса космічного корабля, вартість систем, необхідна кількість екіпажу. Її структура зображена в табл. 3.6.

Таблиця 3.6 – Структура таблиці “sm”

Найменування	Ідентифікатор	Тип поля	Примітки
1	2	3	4
Код модифікатора розміру	id	int	PK, NN
Маса космічного корабля	spaceship_mass	int	U, NN
Довжина космічного корабля	spaceship_length	int	U, NN
Очки життя космічного корабля	dhp	int	U, NN
Базовий модифікатор керованості	handling	int	NN

Таблиця 3.6 – Продовження

Базовий рейтинг стабільності	stability	int	NN
Вартість маскування	stealth_cost	int	U, NN
Вартість динамічного візуального маскування	dynamic_chameleon_cost	int	U, NN

Таблиця “hull_sections” зберігає назви секцій корпусу космічного корабля, таких як ніс, центр та корма, і також назви комбінацій, які використовуються для визначення можливих локацій для встановлення певних систем космічного корабля.

Структура цієї таблиці зображена в табл. 3.7.

Таблиця 3.7 – Структура таблиці “hull_sections”

Найменування атрибутів		Тип поля	Примітки
Код секції	id	int	PK, NN
Назва секції	name	varchar(7)	U, NN

Таблиця “hull_locations” зберігає назви локацій секцій корпусу космічного корабля, від 1 до 6, та “core”. Структура цієї таблиці зображена в табл. 3.8.

Таблиця 3.8 – Структура таблиці “hull_locations”

Найменування атрибутів		Тип поля	Примітки
Код локації	id	int	PK, NN
Назва локації	name	varchar(4)	U, NN

Таблиця “hull_slots” зберігає інформацію про слоти для систем космічного корабля, які належать певній секції та локації.

Ця таблиця пов'язана зовнішніми ключами з таблицею “spaceships” за id-кодом космічного корабля, таблицею “ship_sections” за id-кодом секції, та “hull_locations” за id-кодом локації. Кожному слоту космічного корабля відповідає одна система, зв'язок з якими реалізовано з використанням допоміжних таблиць.

Структура цієї таблиці зображена в табл. 3.9.

Таблиця 3.9 – Структура таблиці “hull_slots”

Найменування атрибутів		Тип поля	Примітки
Код локації	id	int	PK, NN
Код космічного корабля	ship	int	U, NN
Код секції корпусу	section	int	FK(hull_sections.id), NN
Код локації	location	int	FK(hull_locations.id), NN

Таблиця “automation_levels” зберігає дані про види автоматизації космічного корабля, їхні ефекти, та обмеження по технологічному рівню та розміру космічного корабля. Її структура зображена в Додатку Б.

Таблиця “wings” зберігає дані про вартість додавання крил до космічного корабля. Ці дані було винесено в окрему таблицю, так як не всі розміри космічних кораблів можуть мати крила. Структура цієї таблиці зображена в Додатку Б.

Таблиця “spin_gravity” зберігає дані про вартість додавання системи обертальної гравітації. Ці дані було винесено в окрему таблицю, так як не всі розміри космічних кораблів можуть мати таку систему. Структура цієї таблиці зображена в Додатку Б.

Таблиця “ship_conditions” зберігає дані про стан космічного корабля. від якого залежить надійність та вартість. Структура цієї таблиці зображена в Додатку Б.

Таблиця “engine_rooms” зберігає інформацію про інженерні кімнати, таблиця “engine_rooms_to_slots” зберігає інформацію про наявність в слоті інженерної кімнати. Зв’язки між цими таблицями та таблицею “hull_slots” зображені на рис. 3.2. Структура цих таблиць зображена в Додатку Б.

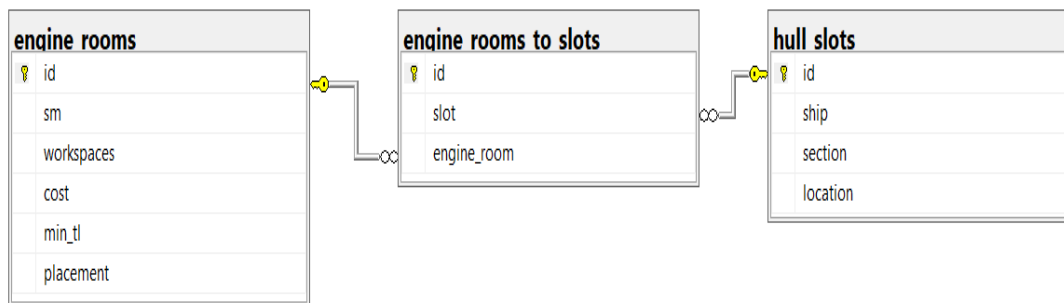


Рисунок 3.2 – Загальний зв’язок системи та слота

Таблиця “external_clamps” зберігає інформацію про системи стикування, таблиця “external_clamps_to_slots” зберігає інформацію наявність в слоті системи стикування. Зв’язки між цими таблицями та таблицею “hull_slots” подібні рис. 3.2. Структура цих таблиць зображена в Додатку Б.

Таблиця “factories” зберігає інформацію про виробничі системи, таблиця “factory_types” зберігає інформацію про типи виробничих систем, і таблиця “factories_to_slots” зберігає інформацію про наявність в слоті виробничої системи.

Зв’язки між цими таблицями та таблицею “hull_slots” зображені на рис. 3.3. Структура цих таблиць зображена в Додатку Б.

Таблиця “force_screen_types” зберігає інформацію про види силових щитів, таблиця “force_screens” зберігає інформацію про силові щити, таблиця “force_screens_to_slots” зберігає інформацію про наявність в слоті силового щита.

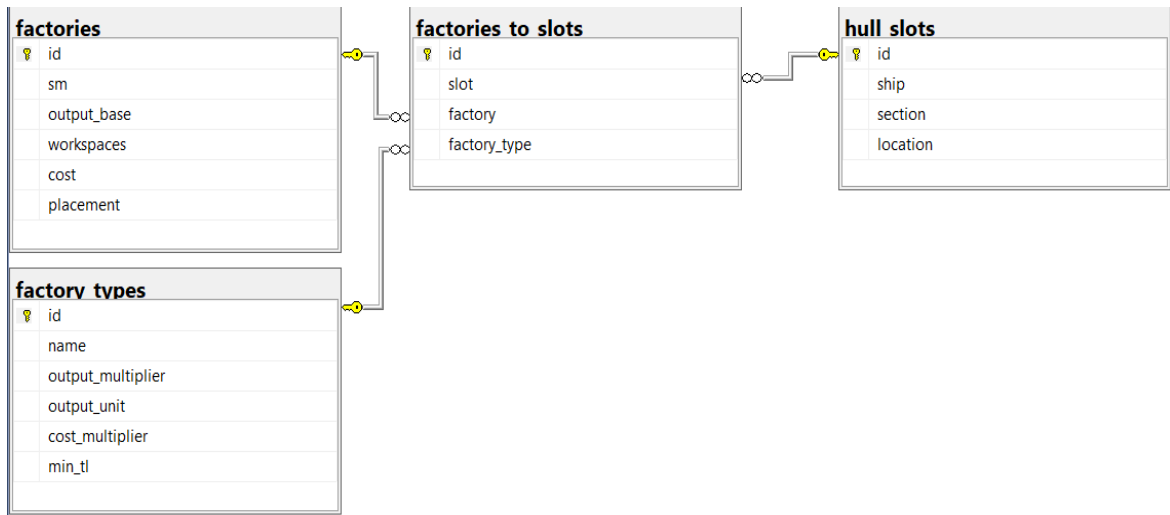


Рисунок 3.3 – Зв’язок системи, типу, та слота

Зв’язки цих таблиць та таблиці “hull_slots” зображені на рис. 3.4. Структура цих таблиць зображена в Додатку Б.

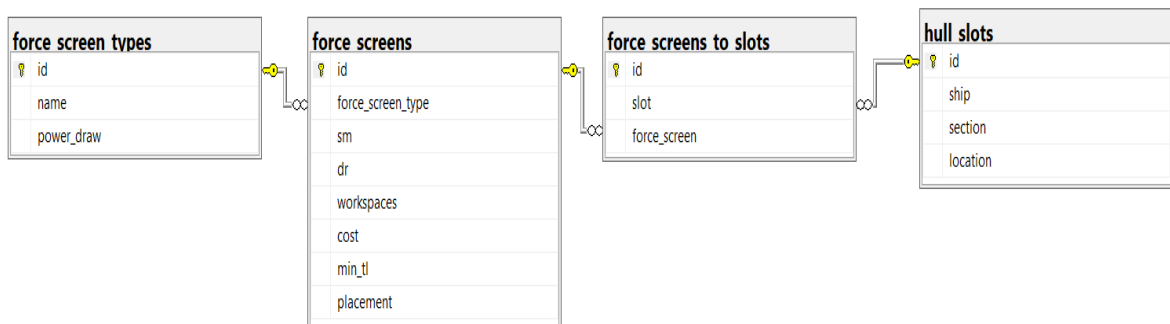


Рисунок 3.4 – Послідовний зв’язок типу, системи, та слотів

Таблиця “stardrive_engines” зберігає інформацію про надсвітлові двигуни, таблиця “stardrive_engine_types” зберігає інформацію про типи надсвітлових двигунів, таблиця “stardrive_engine_categories” зберігає інформацію категорії надсвітлових двигунів, і таблиця “stardrive_engines_to_slots” зберігає інформацію про наявність в слоті надсвітлового двигуна. Зв’язки між цими таблицями та таблицею “hull_slots” зображені на рис. 3.5. Їхня структура зображена в Додатку Б.

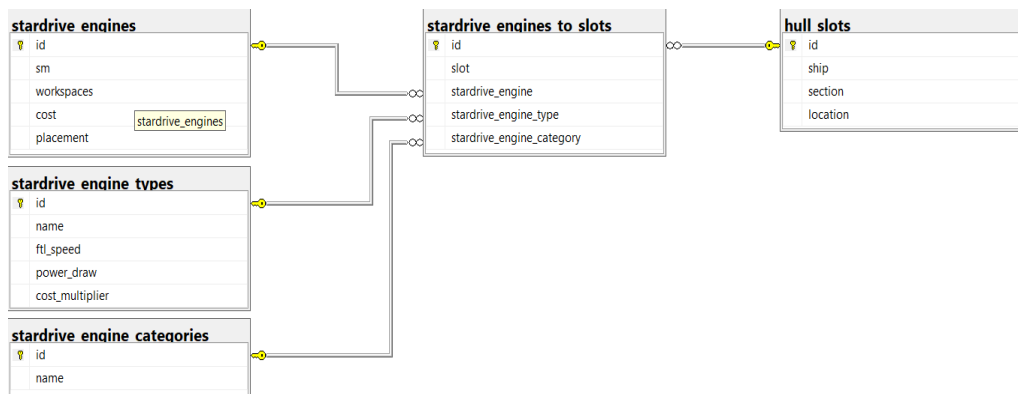


Рисунок 3.5 – Зв’язок системи, типу, категорії, та слотів

Таблиця “jump_gates” зберігає інформацію про телепортаційні системи, таблиця “jump_gates_to_slots” зберігає інформацію наявності в слоті телепортаційної системи. Зв’язки між цими таблицями та таблицею “hull_slots” подібні рис. 3.2. Структура цих таблиць зображена в Додатку Б.

Таблиця “mining_refinery_types” зберігає інформацію про типи шахтарських та переробних систем, таблиця “mining_refinery_systems” зберігає інформацію про шахтарські та переробні системи, і таблиця “mining_refinery_systems_to_slots” зберігає інформацію наявності в слоті шахтарської чи переробної системи. Зв’язок цих таблиць та таблиці “hull_slots” подібні рис. 3.4. Структура цих таблиць зображена в Додатку Б.

Таблиця “open_spaces” зберігає інформацію про системи з відкритим простором, таблиця “open_spaces_to_slots” зберігає інформацію наявності в слоті системи з відкритим простором. Зв’язки між ними та з таблицею “hull_slots” подібні рис. 3.2. Структура цих таблиць зображена в Додатку Б.

Таблиця “robot_arms” зберігає інформацію про зовнішні маніпулятори, таблиця “robot_arms_to_slots” зберігає інформацію наявності в слоті системи з відкритим простором. Зв’язки між ними та з таблицею “hull_slots” подібні рис. 3.2. Структура цих таблиць зображена в Додатку Б.

Таблиця “soft_landing_systems” зберігає інформацію про системи посадки, таблиця “soft_landing_systems_to_slots” зберігає інформацію

наявність в слоті системи посадки. Зв'язки між ними та з таблицею “hull_slots” подібні рис. 3.2. Структура цих таблиць зображена в Додатку Б.

Таблиця “armor_types” зберігає інформацію про типи броні, таблиця “armor_systems” зберігає інформацію про системи броні, таблиця “armor_to_slots” зберігає інформацію про наявність в слоті системи броні. Зв'язки між ними та з таблицею “hull_slots” подібні рис. 3.5. Структура цих таблиць зображена в Додатку Б.

Таблиця “sensor_arrays” зберігає інформацію про сенсорні системи, таблиця “sensor_array_types” зберігає інформацію про типи сенсорних систем, таблиця “sensor_arrays_to_slots” зберігає інформацію про наявність в слоті сенсорної системи. Зв'язки між ними та з таблицею “hull_slots” подібні рис 3.3. Структура цих таблиць зображена в Додатку Б.

Таблиця “power_plant_types” зберігає інформацію про типи силових установок, таблиця “power_plant_variants” зберігає інформацію про варіації силових установок, таблиця “power_plants” зберігає інформацію про силові установки, таблиця “power_plants_to_slots” зберігає інформацію про наявність в слоті силової установки та її варіації. Зв'язки між ними та з таблицею “hull_slots” подібні рис. 3.6. Структура цих таблиць зображена в Додатку Б.

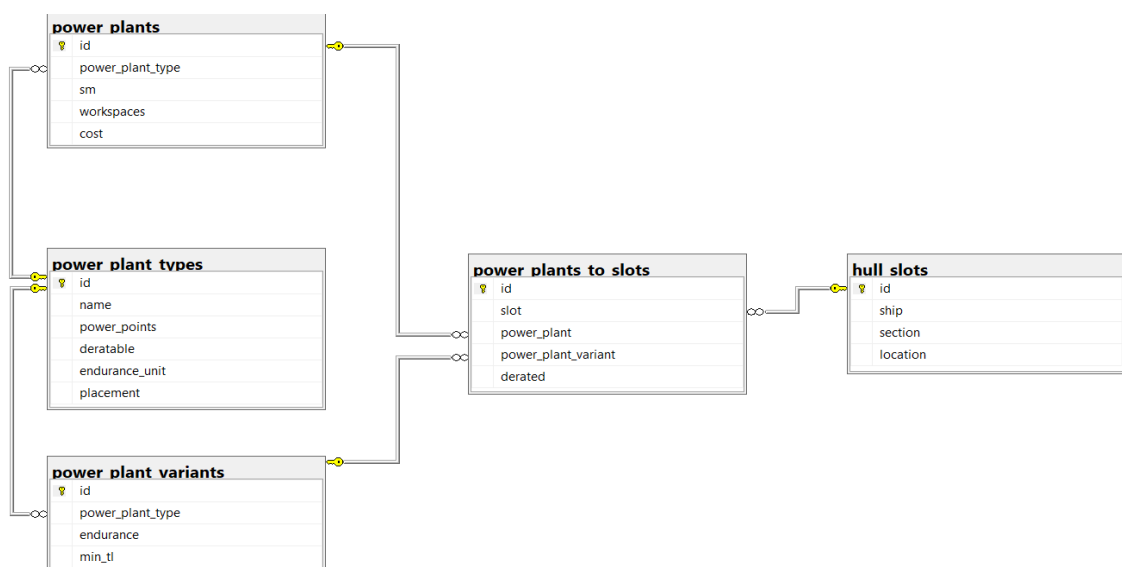


Рисунок 3.6 – Послідовний зв’язок типу з системою та варіантом, та зі слотами

Таблиця “stasis_webs” зберігає інформацію про системи стазису, таблиця “stasis_webs_to_slots” зберігає інформацію наявності в слоті системи стазису.

Зв’язки між ними та з таблицею “hull_slots” подібні рис. 3.2. Структура цих таблиць зображена в Додатку Б.

Таблиця “defensive_ecms” зберігає інформацію про системи електронних контрзаходів, таблиця “defensive_ecms_to_slots” зберігає інформацію наявності в слоті системи електронних контрзаходів.

Зв’язки між ними та з таблицею “hull_slots” подібні рис. 3.2. Структура цих таблиць зображена в Додатку Б.

Таблиця “control_rooms” зберігає інформацію про контрольні системи (містки), таблиця “control_rooms_to_slots” зберігає інформацію наявності в слоті контрольної системи.

Зв’язки між ними та з таблицею “hull_slots” подібні рис. 3.2.

Структура цих таблиць зображена в Додатку Б.

Таблиця “contragravity_lifters” зберігає інформацію про контргравітаційні системи, таблиця “contragravity_lifters_to_slots” зберігає інформацію наявності в слоті контргравітаційної системи.

Зв’язки між ними та з таблицею “hull_slots” подібні рис. 3.2.

Структура цих таблиць зображена в Додатку Б.

Таблиця “cloaking_devices” зберігає інформацію про маскувальні системи, таблиця “cloaking_devices_to_slots” зберігає інформацію наявності в слоті маскувальної системи.

Зв’язки між ними та з таблицею “hull_slots” подібні рис. 3.2.

Структура цих таблиць зображена в Додатку Б.

Таблиця “habitats” зберігає інформацію про житлові відсіки, таблиця “life_support” зберігає інформацію про типи систем забезпечення життєдіяльності, таблиця “habitats_to_slots” зберігає інформацію про наявності в слоті жилого відсіка.

Таблиця “rooms” зберігає інформацію про типи кімнат, таблиця “rooms_to_habitats” зберігає інформацію про кількість кімнат в жилому відсіку. Зв’язки між ними та з таблицею “hull_slots” подібні рис. 3.7. Структура цих таблиць зображена в Додатку Б.

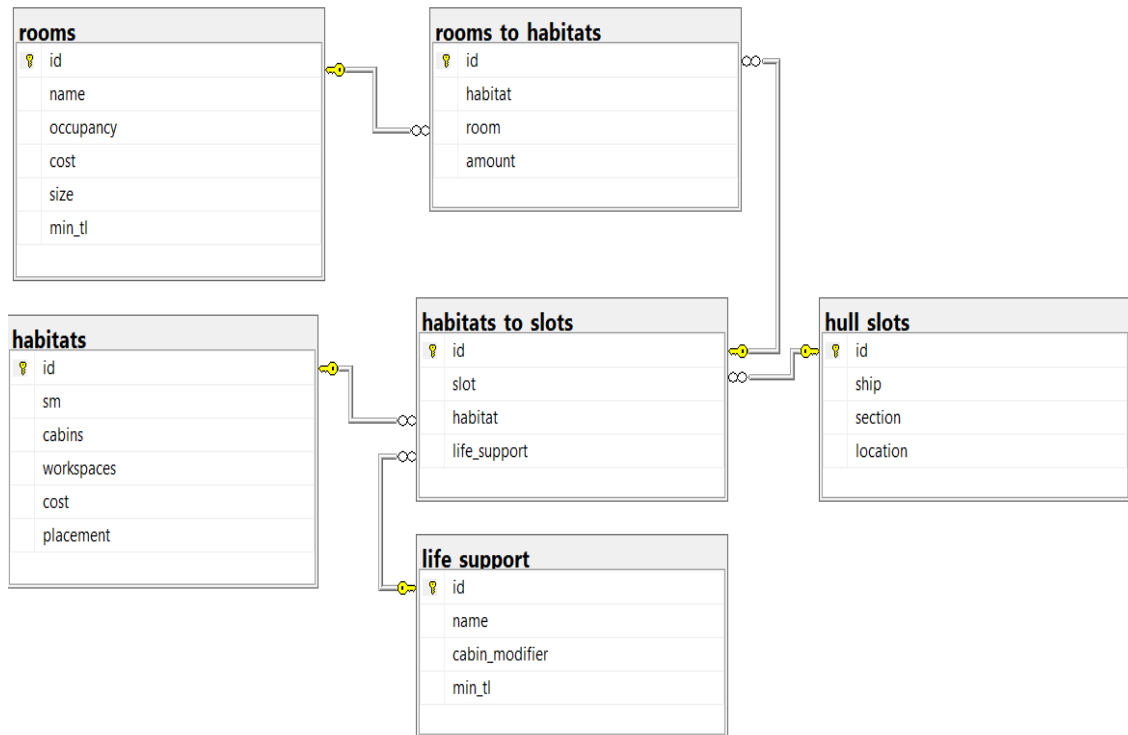


Рисунок 3.7 – Зв’язок житлових відсіків та кімнат зі слотами

Таблиця “fuel_tanks” зберігає інформацію про паливні баки, таблиця “fuel_tanks_to_slots” зберігає інформацію про наявність в слоті паливного баку та поточний вид палива.

Зв’язки між ними та з таблицею “hull_slots” подібні рис. 3.3.

Структура цих таблиць зображена в Додатку Б.

Таблиця “fuel” зберігає інформацію про види палива, таку як його назву та вартість за тону. Її структура зображена в Додатку Б.

Таблиця “engine_types” зберігає інформацію про типи двигунів, таблиця “engine_variations” зберігає інформацію про варіації двигунів, таблиця “engines” зберігає інформацію про двигуни, таблиця “engines_to_slots” зберігає інформацію про наявність в слоті двигуна та його варіації.

Зв'язки між ними та з таблицею “hull_slots” подібні рис. 3.8.

Структура цих таблиць зображена в Додатку Б.

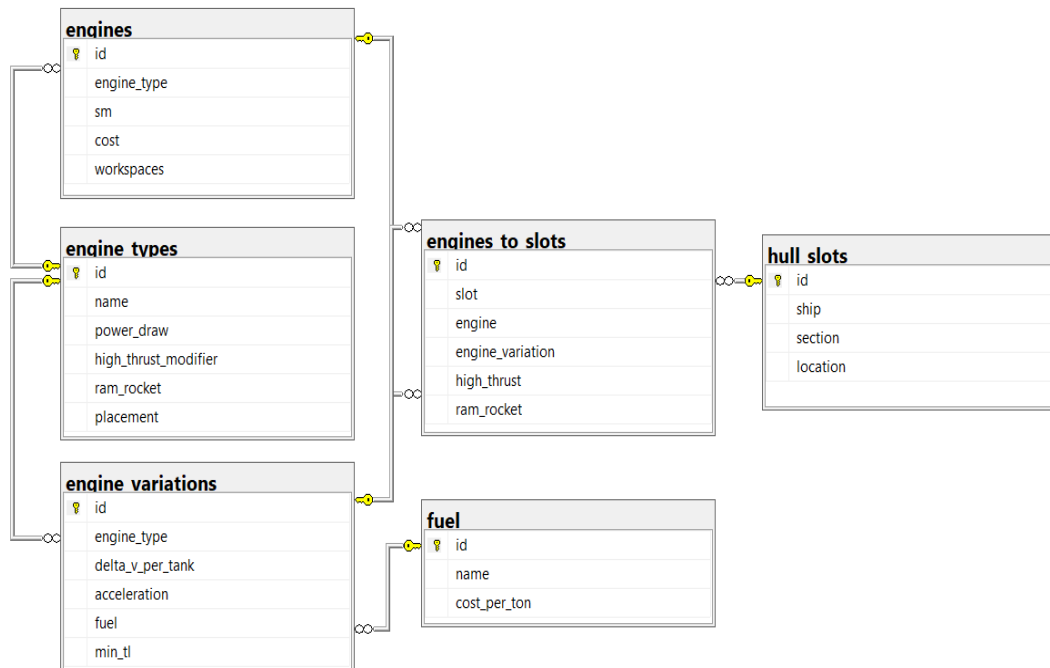


Рисунок 3.8 – Палива з двигунами та двигунів зі слотами

Таблиця “hangar_bays” зберігає інформацію про ангари, таблиця “hangar_bays_to_slots” зберігає інформацію наявності в слоті ангару. Зв’язки між ними та з таблицею “hull_slots” подібні рис. 3.3. Структура цих таблиць зображена в Додатку Б.

Таблиця “cargo_holds” зберігає інформацію про трюми, таблиця “cargo_holds_to_slots” зберігає інформацію наявності в слоті трюму. Зв’язки між ними та з таблицею “hull_slots” подібні рис. 3.2. Структура цих таблиць зображена в Додатку Б.

Таблиця “ramscoops” зберігає інформацію про магнітні ковші, таблиця “ramscoops_to_slots” зберігає інформацію наявності в слоті трюму. Зв’язки між ними та з таблицею “hull_slots” подібні рис. 3.2. Структура цих таблиць зображена в Додатку Б.

Таблиця “passenger_seating” зберігає інформацію про пасажирські сидіння, таблиця “passenger_seating_to_slots” зберігає інформацію наявності в

слоті пасажирських сидінь. Зв'язки між ними та з таблицею “hull_slots” подібні рис. 3.2. Структура цих таблиць зображена в Додатку Б.

Таблиця “weapon_types” зберігає інформацію про типи озброєння, таблиця “weapon_subtypes” зберігає інформацію про підтипи озброєння, таблиця “weapon_placements” зберігає інформацію про види розміщення озброєння, таблиця “rates_of_fire” зберігає інформацію про рівні скорострільності.

Таблиця “weapon_system_types” зберігає інформацію про типи систем озброєння, таблиця “weapon_calibers” зберігає інформацію про калібри, таблиця “weapon_power” зберігає інформацію про потужність зброї, таблиця “beam_ranges” зберігає інформацію про далекобійність енергетичної зброї, таблиця “weapon_system_configurations” зберігає інформацію про конфігурації систем озброєння, таблиця “weapon_systems” зберігає інформацію про системи озброєння, таблиця “weapon_systems_to_slots” зберігає інформацію про наявність в слоті системи озброєння.

Зв'язки між ними та з таблицею “hull_slots” подібні рис. 3.8.

Структура цих таблиць зображена в Додатку Б.

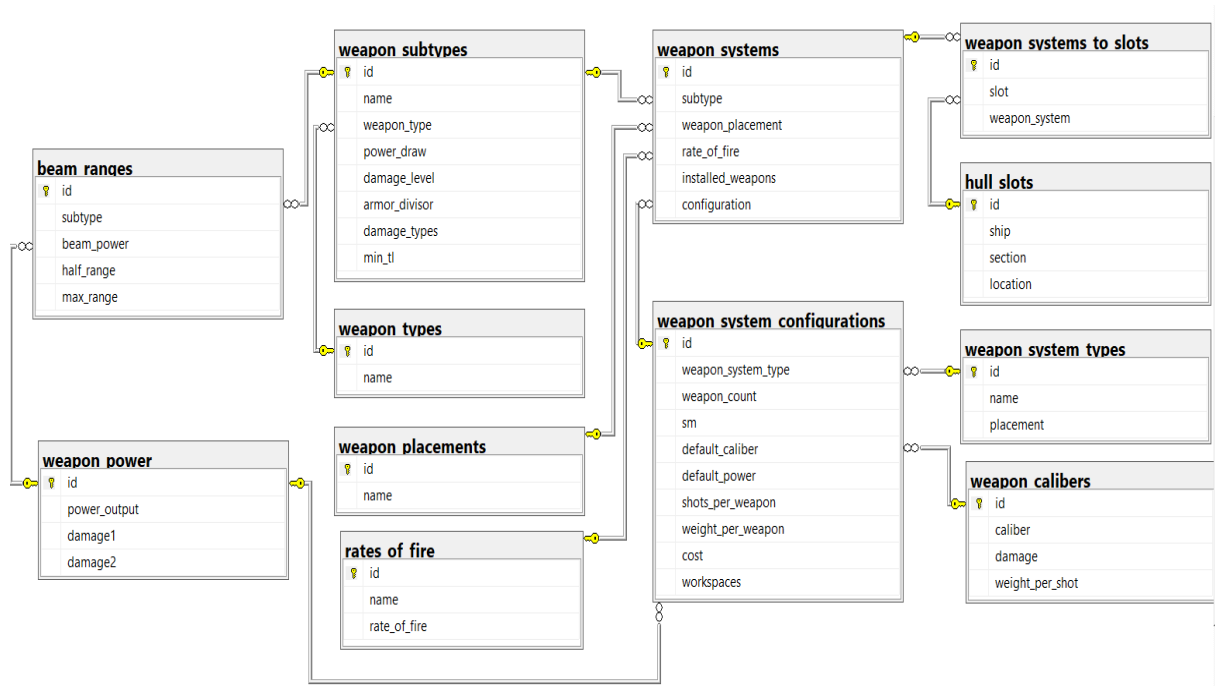


Рисунок 3.8 – Зв'язок систем озброєння зі слотами

3.3 Опис основних функцій інтелектуального помічника

При запуску клієнтської частини інтелектуального помічника “Captain’s Companion”, користувачу доступні чотири опції (рис. 3.9) – перейти до блоку орбітальних маневрів за допомогою кнопки “Navigation”, перейти до блоку космічних кораблів за допомогою кнопки “Spaceship Designer”, перейти до блоку зоряних систем за допомогою кнопки “Star System Charts”, або вийти з програмного забезпечення, натиснувши кнопку “Exit”.

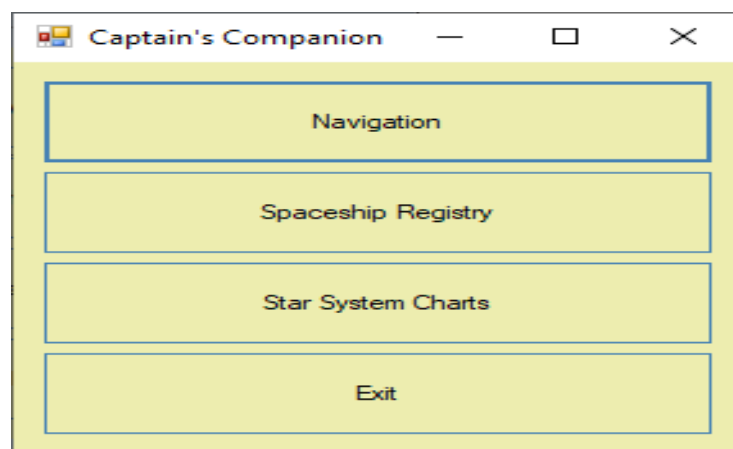


Рисунок 3.9 – Стартове вікно інтелектуального помічника

При запуску клієнтської частини, до відображення цього вікна, відбувається перевірка зв'язку з базою даних. Якщо зв'язок встановити не вдається, користувачу буде виведене відповідне повідомлення з проханням вказати подальші дії (рис. 3.10).

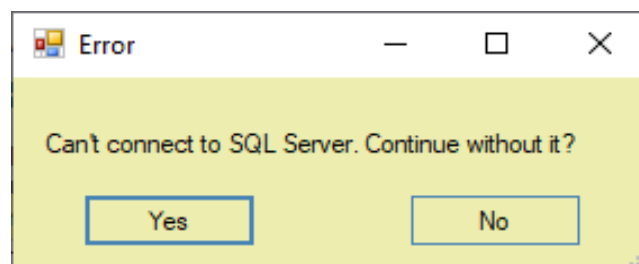


Рисунок 3.10 – Стартове вікно

При відсутності з'єднання користувач може або завершити роботу з інтелектуальним помічником, або продовжити роботу без підключення до бази даних. В такому випадку блоки космічних кораблів та зоряних систем будуть недоступні на цьому вікні (рис. 3.11).

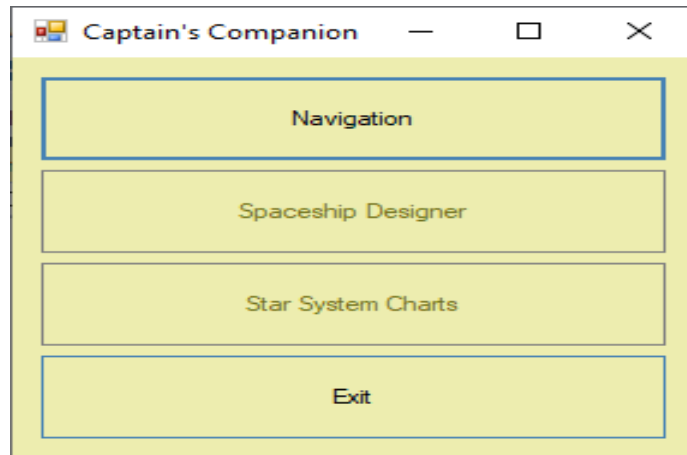


Рисунок 3.11 – Стартове вікно без підключення до БД

При натисканні кнопки “Navigation” на стартовому вікні, користувачу буде доступне вікно “Navigation” (рис. 3.12), яке дозволяє користувачу виконати розрахунки орбітальних маневрів. Користувач може обрати існуючу зоряну систему та небесні тіла, чи ввести дані власноруч. Аналогічно користувач повинен вибрати дані для космічного корабля.

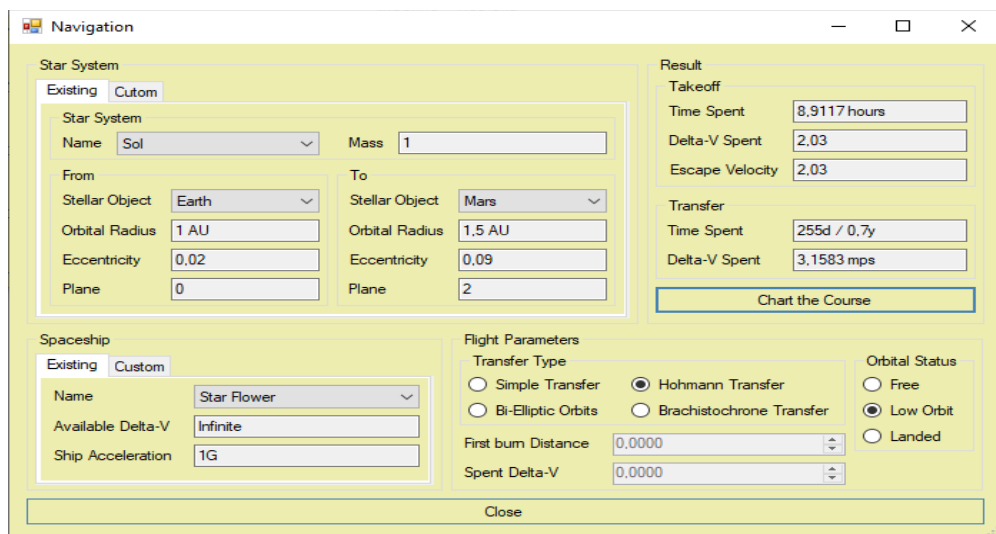


Рисунок 3.12 – Вікно “Navigation”

Після цього, користувач повинен обрати орбітальний маневр з групи “Flight Parameters”. Для деяких маневрів необхідно заповнити поля “First burn Distance” та “Spent Delta-V”, в такому випадку поля стануть доступними для редагування.

Користувач може додатково обрати стан космічного корабля відносно стартової точки - на поверхні планети, на орбіті, чи без впливу.

Коли користувач налаштував всі необхідні параметри, для розрахунку результатів він повинен натиснути кнопку “Chart the Course” в блоці “Result”.

Якщо обраний тип орбітального трансферу є неоптимальним, інтелектуальний помічник виведе користувачу повідомлення про це, з рекомендацією щодо кращого виду трансферу для заданих параметрів.

У випадку використання трансферу, який дозволяє вказати використану кількість палива, при використанні надлишкової кількості, інтелектуальним помічником буде виконано розрахунки з оптимальною кількістю палива, а користувача буде повідомлено про це.

Відповідно до обраних параметрів будуть заповнені поля в цьому блоці. Якщо користувач не підключений до бази даних, вкладки вибору зоряної системи та космічного корабля йому недоступні.

В такому випадку, при відкритті вікна будуть автоматично обраними вкладки ручного введення даних (рис. 3.13).

Користувач може вийти з цього вікна натиснувши кнопку “Close”.

При натисканні кнопки “Spaceship Designer” користувачу буде доступне вікно “Spaceship Registry” (рис. 3.14). В цьому вікні користувач може переглянути космічні кораблі, обравши їхню назву з випадаючого списку.

Згідно з обраною назвою космічного корабля, будуть заповнені поля в групі “View”. Якщо якесь з полів не застосовується до обраного космічного корабля, воно буде мати текст “n/a”.

Рисунок 3.13 – Вікно “Navigation”, ручне введення даних

Одне з полів “Core” завжди буде мати такий текст через особливість системи опису космічних кораблів – космічний корабель може мати лише дві системи, розташовані в слотах “Core”. З інформації про системи космічного корабля, інформаційним помічником будуть розраховані його похідні характеристики та виведені у відповідні поля.

Окрім перегляду інформації про космічні кораблі, користувач може перейти до форм додавання чи редагування космічних кораблів, видалити космічний корабель, чи повернутись до попереднього вікна за допомогою кнопки “Close”.

Для видалення космічного корабля, користувач повинен спочатку обрати його з випадаючого списку “Name” на перегляд, після чого натиснути кнопку “Remove”. Якщо космічний корабель не обраний, користувачу буде виведене повідомлення про необхідність його вибрати.

Після успішного видалення, інформацію про обраний космічний корабель буде прибрано з вікна.

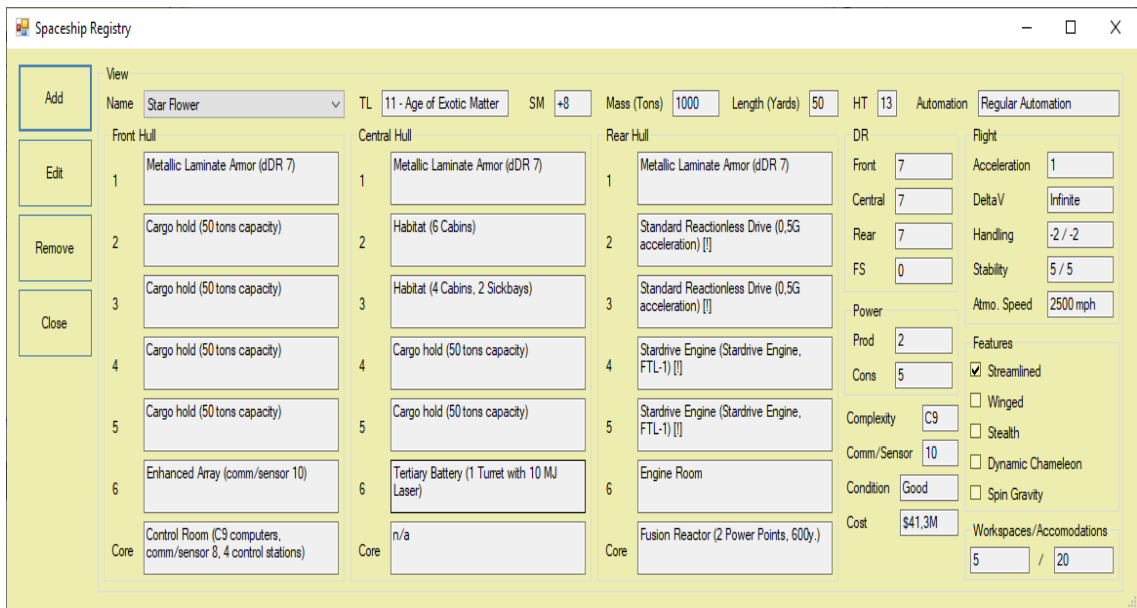


Рисунок 3.14 – Стартове вікно без підключення до БД

Для додавання нового космічного корабля, користувачу необхідно натиснути кнопку “Add”, після чого відкриється вікно “Spaceship Designer”. Це вікно містить три вкладки – “Systems and Features”, “Priority Setup”, та “Additional Setup”.

Вкладка “Systems and Features” (рис. 3.15) є основною вкладкою, на якій користувач може ввести назву космічного корабля, обрати його розмір та технологічний рівень, визначити його стан, рівень автоматизації, аеродинамічність, та інші характеристики, і обрати 20 систем, з яких він складається.

Вибір систем космічного корабля виконується за допомогою випадających списків у лівій частині відповідних блоків, які містять системи, доступні для обраного розміру та технологічного рівня космічного корабля. Додатково, деякі системи мають підтипи, які користувач може обрати з випадającego списку в правій частині блоків, навпроти випадającego списку відповідної системи.

Якщо користувач змінює технологічний рівень чи розмір космічного корабля, інтелектуальним помічником буде додано чи вилучено певні системи чи їхні варіанти з доступних до вибору у випадających списках та вже обраних.

Якщо якась система може бути налаштована з більшою деталізацією ніж можливо за допомогою одного випадуючого списку, вона буде включена до списку в одному з блоків вкладок “Priority Setup” (рис. 3.16) чи “Additional Setup” (рис. 3.17). При внесенні змін в цьому вікні, інтелектуальний помічник буде здійснювати розрахунки похідних характеристик космічного корабля та виводити їх в групу полів “Derived Characteristics”.

The screenshot shows the 'Spaceship Designer' application window with the 'Systems and Features' tab selected. The interface is organized into several functional areas:

- General Characteristics:** Includes input fields for Name, TL (Type), Size, Streamlined (checkbox), Mass (Tons), and Length (Yards).
- Additional Options:** Features dropdowns for Automation and Condition (set to 'Good'), and checkboxes for Winged, Spin Gravity, Stealth, and Dynamic Chameleon.
- Derived Characteristics:** A grid of input fields and checkboxes for various system parameters including Power, Workspaces/Accom, Handling, Stability, Complexity, HT, Atmo. Speed, Cost, Comm/Sensor, and FS.
- Hull Sections:** Three columns representing the Front Hull, Central Hull, and Rear Hull. Each column contains six numbered slots (1-6) with dropdown menus and a 'Core' checkbox at the bottom.
- Buttons:** 'Create' and 'Close' buttons are located at the bottom of the window.

Рисунок 3.15 – Вікно “Spaceship Designer”, вкладка “Systems and Features”

Вкладка “Priority Setup” (рис. 3.16) надає користувачу можливість налаштувати системи озброєння, жилі відсіки, та паливні баки. Системи озброєння та жилі відсіки винесені в цю вкладку через велику кількість доступних параметрів та їхнього впливу, та сильної залежності системи від специфіки обраних опцій.

Паливні баки винесені в цю вкладку через необхідність вибору палива для подальшого виконання розрахунків орбітальних маневрів. Зміни, внесені на цій вкладці, зберігаються лише після натискання кнопки “Apply” у відповідному блоці.

В групі “Weapon Setup” користувач може обрати тип, підтип, та інші характеристики системи озброєння. На основі цих характеристик, інтелектуальним помічником будуть розраховані похідні характеристики та виведені у відповідні поля.

При роботі з жилими відсіками в групі “Habitat Setup” користувач може обрати необхідну йому кількість кімнат та їхній тип. Додавання та видалення кімнат відбувається за допомогою введення числових значень навпроти назв кімнат та натискання кнопок “+” чи “-”. Інтелектуальний помічник не дозволить користувачу додати більше кімнат ніж можливо.

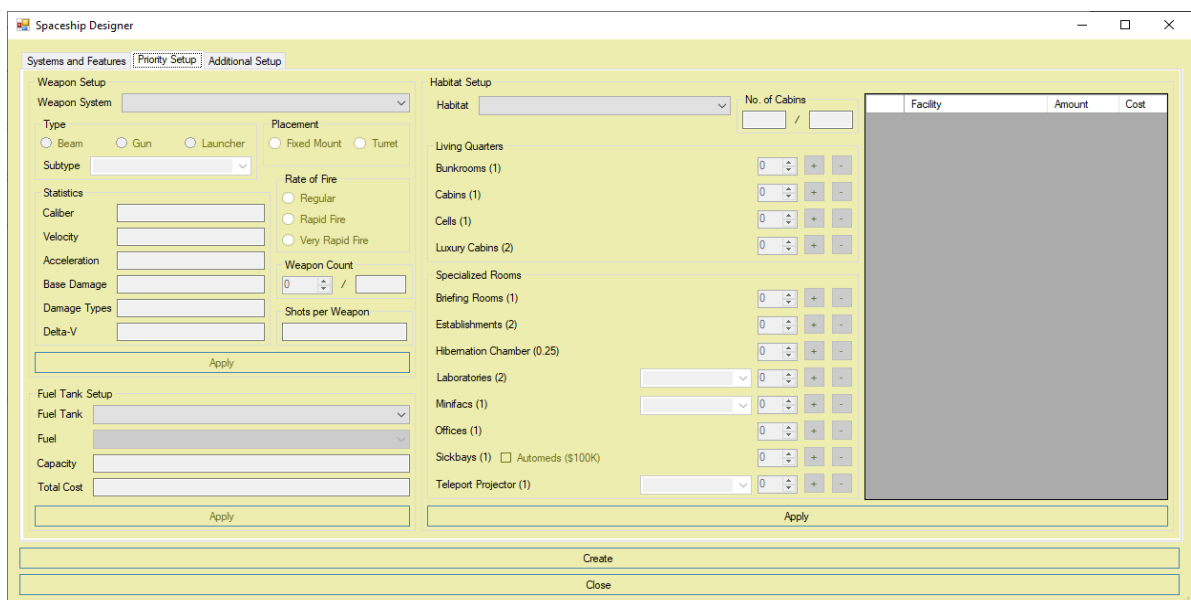


Рисунок 3.16 – Вікно “Spaceship Designer”, вкладка “Priority Setup”

Вкладка “Additional Setup” (рис. 3.17) надає користувачу можливість налаштувати системи броні, пасажирських сидінь, контрольних кімнат (містків), надсвітлових двигунів, силових установок, ангарів, трюмів, та двигунів.

Для внесення змін, користувачу необхідно обрати систему з випадального списку у відповідній групі, виконати налаштування, та натиснути кнопку “Apply”.

The screenshot shows the 'Spaceship Designer' application window with the 'Additional Setup' tab selected. The window is divided into eight main configuration sections, each with its own set of controls and an 'Apply' button:

- Armor Setup:** Includes 'Armor System' (dropdown), 'Hardening' (slider), and 'Cost' (text box).
- Passenger Seating Setup:** Includes 'Seating System' (dropdown), 'Stretchers' (slider), and 'Seats' (text box).
- Control Room Setup:** Includes 'Control Room' (dropdown), 'Control Stations' (slider), and 'Cost' (text box).
- Stardrive Engine Setup:** Includes 'Stardrive' (dropdown), 'Category' (dropdown), and an 'Add' button.
- Power Plant Setup:** Includes 'Power Plant' (dropdown), 'Derating' (slider), 'Power' (text box), 'Endurance' (text box), and 'Cost' (text box).
- Hangar Setup:** Includes 'Hangar' (dropdown), 'Optimized' (checkbox), 'General Launch Rate' (text box), and 'Optimized Launch Rate' (text box).
- Cargo Setup:** Includes 'Cargo Hold' (dropdown), 'Refrigerated' (slider), 'Shielded' (slider), 'Capacity' (text box), and 'Cost' (text box).
- Engine Setup:** Includes 'Engine System' (dropdown), 'Fuel' (dropdown), 'Delta-V' (text box), 'High Thrust' (checkbox), 'Acceleration' (text box), 'Ram-Rocket' (checkbox), and 'Cost' (text box).

At the bottom of the window are two large buttons: 'Create' and 'Close'.

Рисунок 3.17 – Вікно “Spaceship Designer”, вкладка “Additional Setup”

Після внесення всіх необхідних даних, користувач повинен натиснути кнопку “Create” для створення нового космічного корабля. Якщо назва космічного корабля не співпадає з вже існуючою, заповнено 20 систем, та під час заповнення не порушена логіка заповнення, космічний корабель буде створено.

Інтелектуальний помічник виконує перевірку на коректність заповнення даних користувачем. Якщо створення космічного корабля неможливе через допущення користувачем помилки, користувачу буде виведено повідомлення про необхідність її виправлення. Користувач може скористатись кнопкою “Close” для виходу з цього вікна без збереження даних.

Внесення змін до існуючого космічного корабля відбувається після вибору користувачем космічного корабля на перегляд у вікні “Spaceship Registry” та натискання кнопки “Edit”.

Якщо космічний корабель було обрано, відкриється вікно “Spaceship Designer – Edit”. Якщо його обрано не було, користувачу буде виведено повідомлення про необхідність вибору космічного корабля. При відкриття

вікна “Spaceship Designer – Edit”, його поля будуть автоматично заповнені згідно характеристик обраного космічного корабля (рис. 3.18).

The screenshot shows the 'Spaceship Designer - Edit' window with the 'Systems and Features' tab selected. The window is organized into several sections:

- General Characteristics:** Includes fields for Name (Star Rover), TL (11 - Age of Exotic M.), Mass (Tons) (1000), Size (+8), Length (Yards) (50), and a Streamlined checkbox.
- Additional Options:** Includes Automation (Regular Autor), Condition (Good), Spin Gravity, Stealth, Dynamic Chameleon, and Complexity (n/a).
- Derived Characteristics:** Includes Workspaces/Accom (5 / 20), Handling (-2 / -2), Stability (5 / 5), HT (13), Atmo. Speed (2500 mph), Cost (\$41.3 M), and FS (0).
- Hull Sections:** The window is divided into three columns for Front Hull, Central Hull, and Rear Hull. Each column has a list of components (1-6) with dropdown menus for selection. For example, the Front Hull includes Armor, Cargo Hold, and Comm/Sensor Array. The Central Hull includes Armor, Habitat, Cargo Hold, and Weapon Battery. The Rear Hull includes Armor, Engine, Stardrive Engine, and Engine Room.

At the bottom of the window, there are two buttons: 'Save' and 'Close'.

Рисунок 3.18 – Вікно “Spaceship Designer – Edit”, вкладка “Systems and Features”

Елементи керування в цьому вікні подібні до вікна створення космічних кораблів, за винятком кнопки збереження космічного корабля. Назва цієї кнопки змінена на “Save”. Для виходу без збереження, користувач може натиснути кнопку “Close”.

Після натискання кнопки “Star System Charts” на стартовому вікні, користувачу буде доступне вікно “Charts” (рис. 3.19).

В цьому вікні користувач може отримати текстовий опис зоряної системи, обравши її з випадаючого списку “Star System Name”, переглянути графічне представлення системи натиснувши кнопку “View”, створити нову зоряну систему натиснувши кнопку “Create”, змінити існуючу систему натиснувши кнопку “Edit”, видалити зоряну систему за допомогою кнопки “Delete”, чи повернутись до попереднього вікна використовуючи кнопку “Close”.

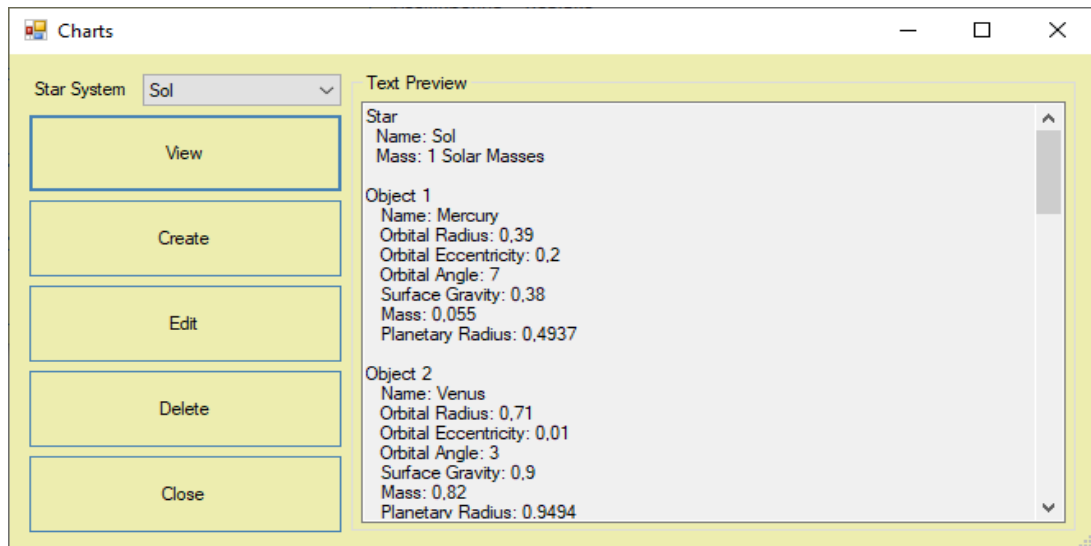


Рисунок 3.19 – Вікно “Charts”

Вікно додавання зоряних систем “Star System – Create” (рис. 3.20) дозволяє користувачу створити нові зоряні системи. Для створення зоряної системи користувач повинен ввести назву зоряної системи, масу зірки, та небесних тіл, які знаходяться на орбіті зірки, після чого натиснути кнопку “Create”.

Якщо назва зоряної системи відсутня серед існуючих, її буде створено та це вікно закриється. Для додавання небесних тіл користувач повинен ввести дані в групі “New Celestial Body” та натиснути кнопку “Add”, після чого запис про небесне тіло з’явиться в таблиці. Всі записи в таблиці відсортовані за їхнім радіусом орбіти навколо зірки.

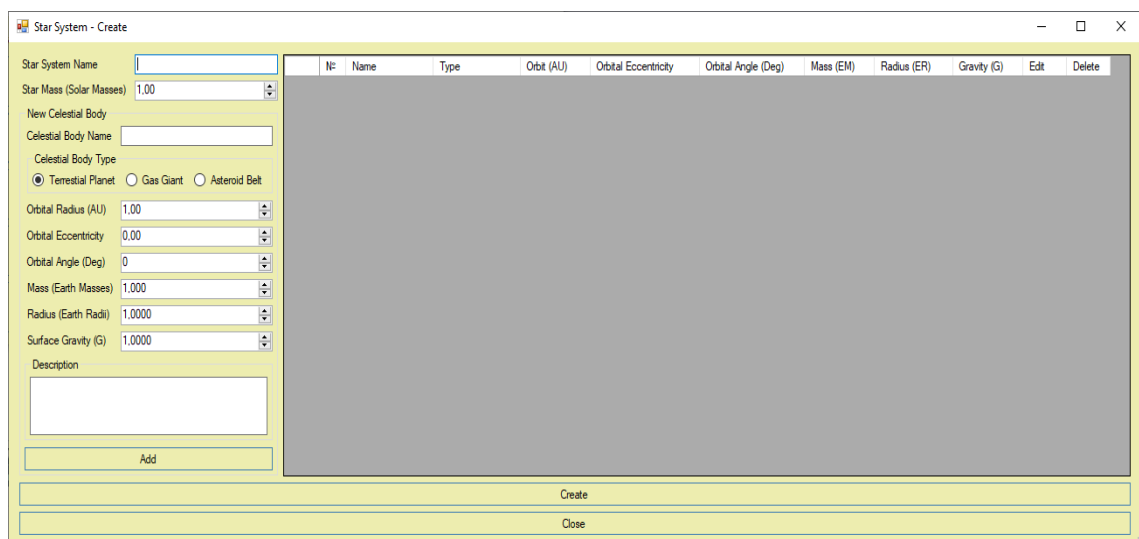


Рисунок 3.20 – Вікно “Star System – Create”

Для внесення змін в додане небесне тіло, користувач може скористатись кнопкою “Edit” у відповідному рядку таблиці. Після натискання на цю кнопку, блок “New Celestial Body” буде перейменований на “Editing - [назва небесного тіла]”, а його поля будуть заповнені відповідною інформацією (рис. 3.21). Користувач може зберегти зміни за допомогою кнопки “Save” чи припинити редагування без збереження кнопкою “Cancel”.

Для видалення небесного тіла, користувачу необхідно натиснути кнопку “Delete” навпроти обраного небесного тіла. Якщо користувачу потрібно вийти з цього вікна без збереження зоряної системи, він може скористатись кнопкою “Close”.

Після натискання кнопки “Edit” у вікні “Charts”, якщо з випадаючого списку була обрана зоряна система, користувачу буде відкрите вікно редагування “Star System - [назва] - Edit”. Це вікно подібне до вікна створення зоряних систем, яке після відкриття буде заповнене даними обраної зоряної системи (рис. 3.21).

Користувач може внести необхідні зміни та натиснути кнопку “Save”. Якщо логіка заповнення даних не порушена та назва зоряної системи не повторюється серед існуючих, зоряну систему буде збережено та закрито це вікно. Якщо логіка заповнення порушена, користувачу буде виведене відповідне повідомлення. Для виходу з вікна без збереження необхідно натиснути кнопку “Close”.

Nr	Name	Type	Orbit (AU)	Orbital Eccentricity	Orbital Angle (Deg)	Mass (EM)	Radius (ER)	Gravity (G)	Edit	Delete
1	Mercury	Terrestrial Planet	0.39	0.2	7	0.055	0.4937	0.38	Edit	Delete
2	Venus	Terrestrial Planet	0.71	0.01	3	0.82	0.9494	0.9	Edit	Delete
3	Earth	Terrestrial Planet	1	0.02	0	1	1	1	Edit	Delete

Рисунок 3.21 – Вікно “Star System - Edit”, редагування планети

Для видалення зоряної системи користувач повинен натиснути кнопку “Delete” у вікні “Charts”, попередньо вибравши з випадаючого списку необхідну зоряну систему. Користувачу буде виведене вікно підтвердження видалення (рис. 3.22), після чого, у випадку підтвердження, зоряну систему буде видалено.

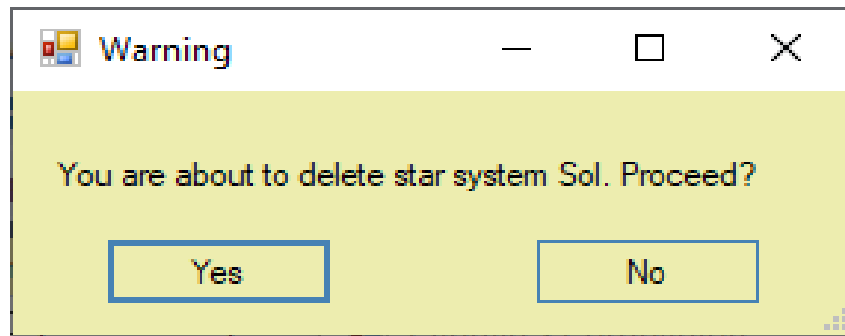


Рисунок 3.22 – Підтвердження видалення зоряної системи

Для перегляду графічного представлення зоряної системи, користувач повинен обрати її з випадаючого списку та натиснути кнопку “View”. Після цього буде відкрите вікно інтелектуального помічника “Viewing - [назва]” з двома вкладками – “Top View” та “Side View” (рис. 3.23). Вкладка “Top View” дозволяє користувачу краще побачити ексцентриситети орбіт небесних тіл, “Side View” – нахил орбітальних площин.

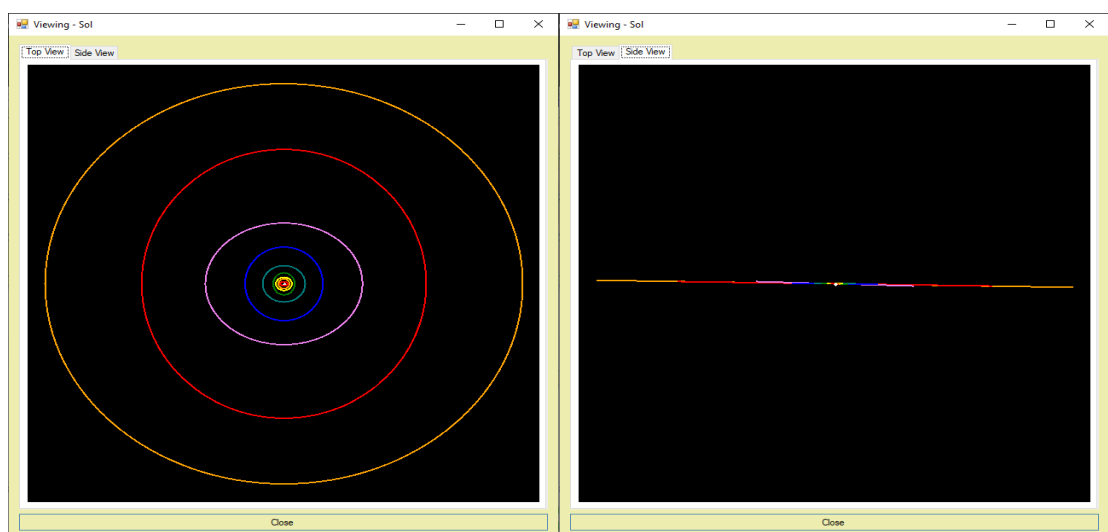


Рисунок 3.23 – Вікно графічного представлення зоряних систем

ВИСНОВКИ

Під час написання магістерської роботи на тему “Інтелектуальний помічник "Captain's Companion" для гравців розширення настільно-рольової гри GURPS Spaceships” було проаналізовано тематичну літературу щодо цього розширення та ПЗ, яке виконує подібні до інтелектуального помічника функції. В результаті аналізу літератури та ПЗ було визначено потенційну користь від створення інтелектуального помічника, сильні сторони аналогів частин функціоналу інтелектуального помічника, які було враховано при його проектуванні, та потенційні недоліки, яким завдяки проведеному аналізу вдалось запобігти.

Інтелектуальний помічник має клієнт-серверну архітектуру. Клієнтською частиною є застосунок з візуальним інтерфейсом користувача, написаний мовою програмування C# та з використанням фреймворку Windows Forms для реалізації візуального інтерфейсу. За функціонування серверної частини інтелектуального помічника відповідає Microsoft SQL Server, який використовується для зберігання та доступ до створених користувачем даних в базі даних. Зв'язок між застосунком та базою даних забезпечується бібліотекою Microsoft.Data.SqlClient, яка дозволяє надсилати запити мовою T-SQL та отримувати відповіді із бази даних.

Під час тестування інтелектуального помічника були виявлені найбільш імовірні проблеми при роботі з ним користувача. Виникнення таких проблем або виправляється в момент виконання користувачем некоректної дії, або, при необхідності застосування складних перевірок для виявлення проблеми, під час спроби користувача перейти до наступного етапу роботи з інтелектуальним помічником.

Інтелектуальний помічник було створено з можливістю роботи без підключення до бази даних, у випадку її відсутності чи пошкодження. Такий підхід дозволяє користувачу починати роботу з частинами інтелектуального помічника, які не вимагають наявності бази даних, відразу після його запуску.

У разі відсутності бази даних, але за наявності можливості її створення, інтелектуальним помічником буде створено її автоматично. Якщо база даних була пошкодженою, інтелектуальний помічник може заново її створити, отримавши на це дозвіл користувача.

Візуальний інтерфейс інтелектуального помічника є простим в навігації, без надлишкової кількості переходів користувача між вікнами одного блоку для виконання необхідних йому дій. Кількість інформаційних вікон з повідомленнями було мінімізовано, використовуючи їх лише для донесення до користувача критичної інформації. Виконання розрахунків в блоці космічних кораблів є автоматичним, не вимагаючи від користувача додаткових дій, що дозволяє бачити в реальному часі як внесені зміни впливають на космічний корабель.

При роботі з інтелектуальним помічником створення зоряних систем, космічних кораблів, та виконання розрахунків набагато пришвидшується, в порівнянні з виконанням цих дій вручну, що свідчить про успішне виконання поставленої мети. Окрім економії часу, виконання розрахунків за допомогою інтелектуального помічника дозволяє усунути можливі помилки, які могла б допустити людина при самостійному їхньому виконанні.

Інтелектуальний помічник має потенціал до подальшого розвитку в декількох напрямках. Одним з напрямків є розширення доступних опцій при створенні космічних кораблів, зоряних систем, та виконанні розрахунків орбітальних маневрів. Іншим напрямком є надання можливості підключення до бази даних декількох користувачів, для одночасної роботи майстра гри та гравців при аналізі доступних маневрів та локацій. Реалізація такого підходу вимагатиме створення додаткових механізмів авторизації та розподілу доступу, дозволяючи гравцям обмежений перегляд інформації, або ж можливість надання різних рівнів доступу різним гравцям з боку майстра гри.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Peter V. Dell'Orto, Sean Punch. GURPS Martial Arts. Steve Jackson Games, Incorporated, 2007.
2. Качура М., Овчарук І. Настільно-рольові ігри та їхні комп'ютерні аналоги. Norwegian Journal of development of the International Science, 2025. № 151. С. 96-99. <https://doi.org/10.5281/zenodo.15536107>
3. Roll a Die. URL: <https://rolladie.net/>
4. Dice Roller - Boring Dice 2.2.0. URL: <https://dice-roller-boring-dice.updatestar.com/en>
5. Steve Jackson Games: Munchkin Level Counter. URL: <https://www.sjgames.com/apps/levelcounter/>
6. Munchkin. URL: <https://munchkin.game/>
2. XCOM: The Board Game – Fantasy Flight Games. URL: <https://www.fantasyflightgames.com/en/products/xcom-the-board-game/products/xcom-core-set/>
3. XCOM: The Board Game | Board Game | BoardGameGeek. URL: <https://boardgamegeek.com/boardgame/163602/xcom-the-board-game>
4. XCOM: TBG App – App Store. URL: <https://apps.apple.com/ua/app/xcom-tbg/id953520552>
5. Качура М.О., Овчарук І.В. Використання програмних застосунків як інтегральних елементів у настільних іграх. Виклики та перспективи розвитку транспортної інфраструктури України: збірник тез III Всеукраїнської науково-практичної конференції (м. Київ, 22 – 24 квітня, 2025). Київ: ДУІТ. С. 516-518.
6. Monopoly Super Electronic Banking Family Board Game with Electronic Banking Unit – Monopoly. URL: <https://shop.hasbro.com/en-us/product/monopoly-super-electronic-banking-family-board-game-with-electronic-banking-unit/E8978>
7. Monopoly Electronic Banking Issues Compared. URL: <https://www.monopolyland.com/monopoly-electronic-banking-editions/>

8. GURPS Character Assistant. URL: <https://www.sjgames.com/gurps/characterassistant/>
9. GURPS 4E Character Builder. URL: <https://jamesturneronline.net/gurps4e-character-builder/>
10. Maps | D&D Virtual Tabletop | D&D Beyond. URL: <https://www.dndbeyond.com/games>
11. GURPS Vehicle Builder. URL: <https://www.sjgames.com/gurps/vehiclebuilder/>
12. David L. Pulver. GURPS Vehicles. Steve Jackson Games, Incorporated, 1998.
13. David L. Pulver. GURPS Spaceships. Steve Jackson Games, Incorporated, 2008.
14. Steve Jackson, Sean Punch, and David L. Pulver. GURPS Basic Set: Characters. Steve Jackson Games, Incorporated, 2004.
15. Steve Jackson, Sean Punch, and David L. Pulver. GURPS Basic Set: Campaigns. Steve Jackson Games, Incorporated, 2004.
16. David L. Pulver. GURPS Spaceships 4: Fighters, Carriers, and Mecha. Steve Jackson Games, Incorporated, 2009. URL: <https://warehouse23.com/products/gurps-spaceships-4-fighters-carriers-and-mecha>
17. David L. Pulver. GURPS Spaceships 7: Divergent and Paranormal Tech. Steve Jackson Games, Incorporated, 2010. URL: <https://warehouse23.com/products/gurps-spaceships-7-divergent-and-paranormal-tech>
18. David L. Pulver. GURPS Spaceships 2: Traders, Liners, and Transports. Steve Jackson Games, Incorporated, 2008. URL: <https://warehouse23.com/products/gurps-spaceships-2-traders-liners-and-transport>
19. David L. Pulver. GURPS Spaceships 3: Warships and Space Pirates. Steve Jackson Games, Incorporated, 2009. URL: <https://warehouse23.com/products/gurps-spaceships-3-warships-and-space-pirates>
20. Ponce T., et al. (2015). Halfway to Anywhere. Pyramid. Space Atlas, 3(79). P. 25-33. URL: <https://warehouse23.com/products/pyramid-number-3-slash-79-space-atlas>

21. Eric's Dominion. GURPS Spaceships Design Spreadsheet. URL: <https://gurpsland.no-ip.org>
22. Pulver D.I., et al. (2011). Mass Combat in Space. Pyramid. Spaceships, 3(30). P. 18-20. URL: <https://warehouse23.com/products/pyramid-number-3-slash-30-spaceships>
23. Pulver D.I., et al. (2011). Alternate Spaceships. Pyramid. Alternate GURPS, 3(34). P. 4-10. URL: <https://warehouse23.com/products/pyramid-number-3-slash-34-alternate-gurps>
24. Peters K., et al. (2012). Mecha Operations. Pyramid. Vehicles, 3(40). P. 4-10. URL: <https://warehouse23.com/products/pyramid-number-3-slash-40-vehicles>
25. GURPS Spaceships Construction Program. URL: <https://spaceship-builder.netlify.app/>
26. GURPS 4th Edition Star System Generator. URL: <https://github.com/tschoppi/starsystem-gen>
27. Jon F. Zeigler, James L. Cambias. GURPS Space. Steve Jackson Games, Incorporated, 2009.
28. Качура М.О. Додаток генерації зоряних систем «Protodisc»: атестаційна робота бакалавра зі спеціальності 121 «Інженерія програмного забезпечення». Київ: ДУІТ, 2024. 68 с.
29. Introduction to C# Windows Forms Applications – GeeksforGeeks. URL: <https://www.geeksforgeeks.org/c-sharp/introduction-to-c-sharp-windows-forms-applications/>
30. Visual Studio: IDE and Code Editor for Software Development. URL: <https://visualstudio.microsoft.com/>
31. Microsoft SQL Server. URL: <https://www.microsoft.com/en-us/sql-server/>
32. NuGet Gallery | Microsoft.Data.SqlClient 6.1.3. URL: <https://www.nuget.org/packages/Microsoft.Data.SqlClient>

ДОДАТКИ

ДОДАТОК А. Структура класів

```

public class StarSystem
{
    public string name;
    public double starMass;
    public List<CelestialBody> celestialBodies;
    public StarSystem(string n, double m)
}

public class CelestialBody
{
    public string name;
    public int type; //1 - terrestrial, 2 gas giant, 3 asteroid
    belt
    public double orbitalRadius;
    public double orbitalEccentricity;
    public int orbitalAngle;
    public double mass;
    public double planetaryRadius;
    public double gravity;
    public string description;
    public CelestialBody (string n, int t, double r, double e,
int a, double m, double p, double g, string d)
}

public class TL
{
    public int number;
    public string name;
    public int complexity;
    public TL()
    public TL(int nu, string na, int c)
}

public class SM
{
    public int modifier;
    public int mass;
    public int length;
    public int dhp;
    public int handling;
    public int stability;
    public int stealthCost;
    public int chameleonCost;
    public SM()
    public SM(int mo, int ma, int l, int d, int h, int s, int
stc, int shc)
}

```

```

public class AutomationLevel
{
    public string name;
    public double crewFactor;
    public int costPerCrew;
    public int minSM;
    public int minTL;
    public AutomationLevel()
    public AutomationLevel(string n, double cf, int cpc, int ms,
int mt)
}
public class Wings
{
    public int sm;
    public int cost;
    public Wings(int s, int c)
}
public class SpinGravity
{
    public int sm;
    public double maxGravity;
    public int cost;
    public SpinGravity(int s, double mg, int c)
}
public class ShipCondition
{
    public string name;
    public double costModifier;
    public int healthModifier;
    public ShipCondition()
    public ShipCondition(string n, double cm, int hm)
}
public class EngineRoom
{
    public int sm;
    public int workspaces;
    public int cost;
    public int minTL;
    public EngineRoom()
    public EngineRoom(int s, int w, int c, int mtl)
}
public class ExternalClamp
{
    public int sm;
    public int cost;
    public int minTL;
    public ExternalClamp()
    public ExternalClamp(int s, int c, int mtl)
}
public class Factory
{
    public int sm;
    public int output;

```

```

        public int workspaces;
        public int cost;
        public Factory()
        public Factory(int s, int o, int w, int c)
    }
public class FactoryType
{
    public string name;
    public double outputMultiplier;
    public string outputUnit;
    public int costMultiplier;
    public int minTL;
    public FactoryType()
    public FactoryType(string n, double om, string ou, int cm,
int mtl)
}
public class CompleteFactory
{
    public Factory factory;
    public FactoryType factoryType;
    public CompleteFactory(Factory f, FactoryType ft)
    public long getCost()
}
public class ForceScreen
{
    public int type;
    public int sm;
    public int dr;
    public int workspaces;
    public int cost;
    public int tl;
    public ForceScreen()
    public ForceScreen(int ty, int s, int d, int w, int c, int
tl)
}
public class ForceScreenType
{
    public int id;
    public string name;
    public int powerDraw;
    public ForceScreenType(int i, string n, int pd)
}
public class JumpGate
{
    public int sm;
    public int maxTonnage;
    public int workspaces;
    public int cost;
    public JumpGate()
    public JumpGate(int s, int mt, int w, int c)
}

```

```

public class MiningRefinery
{
    public int sm;
    public int systemType;
    public double resourceOutput;
    public int workspaces;
    public int cost;
    public MiningRefinery()
    public MiningRefinery(int s, int st, double ro, int w, int
c)
}
public class MiningRefineryType
{
    public int id;
    public string name;
    public MiningRefineryType(int i, string n)
}
public class OpenSpace
{
    public int sm;
    public int areas;
    public int workspaces;
    public int cost;
    public OpenSpace()
    public OpenSpace(int s, int a, int w, int c)
}
public class RobotArm
{
    public int sm;
    public int workspaces;
    public int cost;
    public RobotArm()
    public RobotArm(int s, int w, int c)
}
public class CompleteRobotArm
{
    public RobotArm robotArm;
    public bool cheap;
    public CompleteRobotArm(RobotArm ra, bool c)
    public int getCost()
}
public class SoftLandingSystem
{
    public int sm;
    public SoftLandingSystem()
    public SoftLandingSystem(int s, int c)
}
public class StardriveEngine
{
    public int sm;
    public int workspaces;
    public int cost;
    public StardriveEngine()

```

```

        public StardriveEngine(int s, int w, int c)
    }
public class StardriveEngineType
{
    public string name;
    public int ftlSpeed;
    public int powerDraw;
    public int costMultiplier;
    public StardriveEngineType()
    public StardriveEngineType(string n, int ftls, int pd, int
cm)
}
public class CompleteStardriveEngine
{
    public StardriveEngine stardriveEngine;
    public StardriveEngineType stardriveEngineType;
    public string stardriveEngineCategory;
    public CompleteStardriveEngine(StardriveEngine se,
StardriveEngineType set, string sec)
    public long getCost()
}
public class StasisWeb
{
    public int sm;
    public int workspaces;
    public int cost;
    public StasisWeb()
    public StasisWeb(int s, int w, int c)
}
public class DefensiveECM
{
    public int sm;
    public int workspaces;
    public int cost;
    public int minTL;
    public DefensiveECM()
    public DefensiveECM(int s, int w, int c, int mtl)
}
public class ControlRoom
{
    public int sm;
    public int baseComplexity;
    public int commSensorLevel;
    public int maxControlSations;
    public int controlStationCost;
    public int workspaces;
    public int cost;
    public ControlRoom()
    public ControlRoom(int s, int bc, int csl, int mcs, int csc,
int w, int c)
}

```

```

public class CompleteControlRoom
{
    public ControlRoom controlRoom;
    public int installedControlStations;
    public CompleteControlRoom(ControlRoom cr, int ics)
    public int getCost()
    public int getCost(int ics)
}
public class ContragravityLifter
{
    public int sm;
    public int workspaces;
    public int cost;
    public ContragravityLifter()
    public ContragravityLifter(int s, int w, int c)
}
public class CloakingDevice
{
    public int sm;
    public int workspaces;
    public int cost;
    public CloakingDevice()
    public CloakingDevice(int s, int w, int c)
}
public class Habitat
{
    public int sm;
    public int cabins;
    public int workspaces;
    public int cost;
    public Habitat()
    public Habitat(int s, int ca, int w, int co)
}
public class LifeSupportType
{
    public string name;
    public int cabinModifier;
    public int minTL;
    public LifeSupportType()
    public LifeSupportType(string n, int cm, int mtl)
}
public class Room
{
    public string name;
    public int occupancy;
    public int cost;
    public double size;
    public int minTL;
    public Room()
    public Room(string n, int o, int c, double s, int mtl)
}

```

```

public class RoomCount
{
    public Room room;
    public int count;
    public RoomCount(Room r, int c)
}

public class CompleteHabitat
{
    public Habitat habitat;
    public LifeSupportType lifeSupportType;
    public List<RoomCount> roomCount;
    public CompleteHabitat(Habitat h, LifeSupportType lst)
    public bool addRooms(string roomName, int amount, List<Room>
rooms)
        public double findUnoccupiedSpace()
        public bool removeRooms(string roomName, int amount)
        public double findSteerageCargo()
        public void trimRooms()
        public long getCost()
}

public class FuelTank
{
    public int sm;
    public double capacity;
    public int cost;
    public FuelTank()
    public FuelTank(int s, double ca, int co)
}

public class Fuel
{
    public int id;
    public string name;
    public long costPerTon;
    public Fuel()
    public Fuel(int i, string n, long cpt)
}

public class CompleteFuelTank
{
    public FuelTank fuelTank;
    public Fuel fuel;
    public CompleteFuelTank(FuelTank ft, Fuel f)
}

public class EngineType
{
    public int id;
    public string name;
    public int powerDraw;
    public int highThrustModifier;
    public bool ramRocket;
    public int placement;
    public EngineType()

```

```

        public EngineType(int i, string n, int pd, int htm, bool rr,
int p)
    }
public class EngineVariation
{
    public int id;
    public int engineType;
    public double deltaVPerTank;
    public double acceleration;
    public int fuel;
    public int minTL;
    public EngineVariation()
    public EngineVariation(int i, int et, double dvpt, double a,
int f, int mtl)
}
public class Engine
{
    public int id;
    public int engineType;
    public int sm;
    public int workspaces;
    public int cost;
    public Engine()
    public Engine(int i, int et, int s, int w, int c)
}
public class CompleteEngine
{
    public Engine engine;
    public EngineType engineType;
    public EngineVariation engineVariation;
    public bool highThrust;
    public bool ramRocket;
    public CompleteEngine(Engine e, EngineType et,
EngineVariation ev, bool ht, bool rr)
    public long getCost()
    public double getAcceleration()
}
public class ArmorType
{
    public int id;
    public string name;
    public int minTL;
    public ArmorType(int i, string n, int mtl)
}
public class ArmorSystem
{
    public int sm;
    public int armorType;
    public int streamlinedDR;
    public int unstreamlinedDR;
    public int cost;

```

```

        public ArmorSystem()
        public ArmorSystem(int s, int at, int sdr, int udr, int c)
    }
public class CompleteArmorSystem
{
    public ArmorSystem armorSystem;
    public int hardening;
    public CompleteArmorSystem(ArmorSystem arsy, int h)
    public double getCost()
    public double getCost(int level)
}
public class HangarBay
{
    public int sm;
    public int capacity;
    public int launchRate;
    public int workspaces;
    public int cost;
    public HangarBay()
    public HangarBay(int s, int ca, int lr, int w, int co)
}
public class CompleteHangarBay
{
    public HangarBay hangarBay;
    public int optimization;
    public CompleteHangarBay(HangarBay hb, int o)
}
public class CommSensorArray
{
    public int sm;
    public int arrayLevel;
    public int workspaces;
    public int cost;
    public CommSensorArray()

        public CommSensorArray(int s, int al, int w, int c)
}
public class CommSensorArrayType
{
    public string name;
    public int costMultiplier;
    public CommSensorArrayType()
    public CommSensorArrayType(string n, int cm)
}
public class CompleteCommSensorArray
{
    public CommSensorArray commSensorArray;
    public CommSensorArrayType commSensorArrayType;
    public CompleteCommSensorArray(CommSensorArray csa,
CommSensorArrayType csat)
        public long getCost()
}

```

```

public class CargoHold
{
    public int sm;
    public double capacity;
    public CargoHold()
    public CargoHold(int s, double c)
}
public class CompleteCargoHold
{
    public CargoHold cargoHold;
    public double refrigeratedTonnage;
    public double shieldedTonnage;
    public CompleteCargoHold(CargoHold ch, double rt, double st)
    public double getTonnage()
    public double getCost()
}
public class Ramscoop
{
    public int sm;
    public int workspaces;
    public int cost;
    public Ramscoop()
    public Ramscoop(int s, int w, int c)
}
public class PowerPlantType
{
    public int id;
    public string name;
    public int powerPoints;
    public bool deratable;
    public string enduranceUnit;
    public int placement;
    public PowerPlantType(int i, string n, int pp, bool d,
string eu, int p)
}
public class PowerPlantVariant
{
    public int id;
    public int powerPlantType;
    public int endurance;
    public int minTL;
    public PowerPlantVariant()
    public PowerPlantVariant(int i, int ppt, int e, int mtl)
}
public class PowerPlant
{
    public int id;
    public int powerPlantType;
    public int sm;
    public int workspaces;
    public int cost;
    public PowerPlant()

```

```

        public PowerPlant(int i, int ppt, int s, int w, int c)
    }
    public class CompletePowerPlant
    {
        public PowerPlant powerPlant;
        public PowerPlantVariant powerPlantVariant;
        public int derated;
        public CompletePowerPlant(PowerPlant pp, PowerPlantVariant
ppv, int d)
        public int getPowerPoints(List<PowerPlantType>
powerPlantTypes)
        public string getEndurance(List<PowerPlantType>
powerPlantTypes)
        public double getCost(List<PowerPlantType> powerPlantTypes)
    }
    public class PassengerSeating
    {
        public int sm;
        public int seats;
        public int cost;
        public PassengerSeating()
        public PassengerSeating(int sm, int se, int c)
    }
    public class CompletePassengerSeating
    {
        public PassengerSeating passengerSeating;
        public int stretchers;
        public CompletePassengerSeating(PassengerSeating ps, int s)
    }
    public class WeaponType
    {
        public int id;
        public string name;
        public WeaponType(int i, string n)
    }
    public class WeaponSubtype
    {
        public int id;
        public string name;
        public int weaponType;
        public int powerDraw;
        public int damageLevel;
        public int armorDivisor;
        public string damageTypes;
        public int minTL;
        public WeaponSubtype()
        public WeaponSubtype(int i, string n, int wt, int pd, int
dl, int ad, string dt, int mtl)
    }
    public class WeaponPlacement
    {
        public int id;

```

```

        public string name;
        public WeaponPlacement()
        public WeaponPlacement(int i, string n)
    }
public class RateOfFire
{
    public int id;
    public string name;
    public int rateOfFire;
    public RateOfFire()
    public RateOfFire(int i, string n, int rof)
}
public class WeaponSystemType
{
    public int id;
    public string name;
    public int placement;
    public WeaponSystemType(int i, string n, int p)
}
public class WeaponCaliber
{
    public int id;
    public double caliber;
    public string damage;
    public double weightPerShot;
    public WeaponCaliber(int i, double c, string d, double wps)
}
public class WeaponPower
{
    public int id;
    public long powerOutput;
    public string damage1;
    public string damage2;
    public WeaponPower(int i, long po, string d1, string d2)
}
public class BeamRange
{
    public int subtypeID;
    public int beamPowerID;
    public int halfRange;
    public int maxRange;
    public BeamRange(int sid, int bpid, int hr, int mr)
}
public class WeaponSystemConfiguration
{
    public int id;
    public int weaponSystemType;
    public int weaponCount;
    public int sm;
    public int defaultCaliber;
    public int defaultPower;
    public int shotsPerWeapon;

```

```

    public int weightPerWeapon;
    public int workspaces;
    public int cost;
    public WeaponSystemConfiguration()
    public WeaponSystemConfiguration(int i, int wst, int wc, int
s, int dc, int dp, int spw, int wpw, int w, int c)
}
public class CompleteWeaponSystem
{
    public int id;
    public WeaponSubtype weaponSubtype;
    public WeaponPlacement weaponPlacement;
    public RateOfFire rateOfFire;
    public int installedWeapons;
    public WeaponSystemConfiguration weaponSystemConfiguration;
    public CompleteWeaponSystem(WeaponSubtype ws,
WeaponPlacement wp, RateOfFire rof, int iw,
WeaponSystemConfiguration wsc)
    public CompleteWeaponSystem(int i, WeaponSubtype ws,
WeaponPlacement wp, RateOfFire rof, int iw,
WeaponSystemConfiguration wsc)
    public long getCost()
}
public class Spaceship
{
    public string name;
    public TL tl;
    public SM sm;
    public bool streamlined;
    public Wings wings;
    public SpinGravity spinGravity;
    public bool stealth;
    public bool dynamicChameleon;
    public AutomationLevel automation;
    public ShipCondition condition;
    public object[] systems;
    public int[] coreSystems;
    public Spaceship(string n, TL t, SM s, bool st, Wings w,
SpinGravity sg, bool ste, bool dc, AutomationLevel al,
ShipCondition c, object[] sy, int[] cs)
    public int getHT()
    public double getAirbreathingSpeed()
    private bool checkFuel(int fuelType)
}

```

ДОДАТОК Б. Структури таблиць БД

Таблиця – Структура таблиці “automation_levels”

Найменування	Ідентифікатор	Тип поля	Примітки
Код рівня автоматизації	id	int	PK, NN
Назва рівня автоматизації	name	int	U, NN
Модифікатор екіпажу	crew_factor	int	U, NN
Вартість заміни екіпажу	cost_per_crew	int	NN
Мінімальний розмір космічного корабля	min_sm	int	NN
Мінімальний технологічний рівень	min_tl	int	NN

Таблиця – Структура таблиці “wings”

Найменування	Ідентифікатор	Тип поля	Примітки
Код крил	id	int	PK, NN
Модифікатор розміру	sm	int	FK(sm.id), U, NN
Вартість	cost	int	U, NN

Таблиця – Структура таблиці “spin_gravity”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи обертальної гравітації	id	int	PK, NN
Модифікатор розміру	sm	int	FK(sm.id), U, NN
Максимальний рівень обертальної гравітації	max_g	float	U, NN
Вартість	cost	int	U, NN

Таблиця – Структура таблиці “ship_conditions”

Найменування	Ідентифікатор	Тип поля	Примітки
Код стану	id	int	PK, NN
Назва стану	name	varchar(10)	U, NN
Модифікатор вартості	price_modifier	float	U, NN
Модифікатор надійності	health_modifier	int	U, NN

Таблиця – Структура таблиці “engine_rooms_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Інженерна кімната	engine_room	int	FK(engine_rooms.id), NN

Таблиця – Структура таблиці “engine_rooms”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Кількість екіпажу	workspaces	int	NN
Вартість	cost	int	U, NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “force_screen_types”

Найменування	Ідентифікатор	Тип поля	Примітки
Код типу силового щита	id	int	PK, NN
Назва типу	name	varchar(11)	U, NN
Максимальна потужність	power_draw	int	U, NN

Таблиця – Структура таблиці “force_screens”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Тип силового щита	force_screen_type	int	FK(force_screen_types.id), NN
Розмір	sm	int	FK(sm.id), NN
Рівень захисту	dr	int	NN
Кількість екіпажу	workspaces	int	NN
Вартість	cost	int	NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “force_screens_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Виробнича система	force_screen	int	FK(force_screens.id), NN

Таблиця – Структура таблиці “external_clamps”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Вартість	cost	int	U, NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “external_clamps_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Система стикування	external_clamp	int	FK(external_clamps.id), NN

Таблиця – Структура таблиці “factories”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Базовий об’єм виробництва	output_base	int	U, NN
Кількість екіпажу	workspaces	int	NN
Вартість	cost	int	U, NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “mining_refinery_types”

Найменування	Ідентифікатор	Тип поля	Примітки
Код типу системи	id	int	PK, NN
Назва	name	varchar(8)	U, NN

Таблиця – Структура таблиці “factory_types”

Найменування	Ідентифікатор	Тип поля	Примітки
Код типу виробничої системи	id	int	PK, NN
Назва	name	varchar(11)	U, NN
Модифікатор об'єму виробництва	output_multiplier	float	NN
Одиниця виміру	output_unit	varchar(4)	NN
Модифікатор вартості	cost_multiplier	int	U, NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN

Таблиця – Структура таблиці “factories_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Виробнича система	factory	int	FK(factories.id), NN
Тип системи	factory_type	int	FK(factory_types.id), NN

Таблиця – Структура таблиці “mining_refinery_systems”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Тип системи	system_type	int	FK(mining_refinery_types.id), NN
Виробництво ресурсів	resource_output	float	NN
Кількість екіпажу	workspaces	int	NN
Вартість	cost	int	U, NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “mining_refinery_systems_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Система	mining_refinery_system	int	FK(mining_refinery_systems.id), NN

Таблиця – Структура таблиці “jump_gates”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Максимальний тоннаж	max_tonnage	int	U, NN
Кількість екіпажу	workspaces	int	NN
Вартість	cost	int	U, NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “jump_gates_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Телепортаційна система	jump_gate	int	FK(jump_gates.id), NN

Таблиця – Структура таблиці “open_spaces”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Кількість зон	areas	int	NN, U
Кількість екіпажу	workspaces	int	NN
Вартість	cost	int	U, NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “open_spaces_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Система	open_space	int	FK(open_spaces.id), NN

Таблиця – Структура таблиці “robot_arms”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Кількість екіпажу	workspaces	int	NN
Вартість	cost	int	U, NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “robot_arms_to_slots ”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Маніпулятор	robot_arm	int	FK(robot_arms.id), NN
Низька точність	grabber	bit	NN

Таблиця – Структура таблиці “soft_landing_systems”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Вартість	cost	int	U, NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “soft_landing_systems_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Система	soft_landing_system	int	FK(soft_landing_systems.id), NN

Таблиця – Структура таблиці “armor_types”

Найменування	Ідентифікатор	Тип поля	Примітки
Код типу	id	int	PK, NN
Назва	name	varchar(29)	U, NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN

Таблиця – Структура таблиці “armor_systems”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Тип броні	armor_type	int	FK(armor_types.id), NN
Рівень захисту для аеродинамічних кораблів	streamlined_dr	int	NN
Рівень захисту для неаеродинамічних кораблів	unstreamlined_dr	int	NN
Вартість	cost	int	NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “armor_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Система броні	armor_system	int	FK(armor_systems.id), NN
Зміцнення	hardening	int	NN

Таблиця – Структура таблиці “stardrive_engines”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Кількість екіпажу	workspaces	int	NN
Вартість	cost	int	U, NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “stardrive_engine_types”

Найменування	Ідентифікатор	Тип поля	Примітки
Код типу надсвітлового двигуна	id	int	PK, NN
Назва	name	varchar(22)	U, NN
Надсвітлова швидкість	ftl_speed	int	NN
Максимальна потужність	power_draw	int	NN
Модифікатор вартості	cost_multiplier	int	NN

Таблиця – Структура таблиці “stardrive_engine_categories”

Найменування	Ідентифікатор	Тип поля	Примітки
Код категорії надсвітлового двигуна	id	int	PK, NN
Назва	name	varchar(17)	U, NN

Таблиця – Структура таблиці “stardrive_engines_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Надсвітловий двигун	stardrive_engine	int	FK(stardrive_engines.id), NN
Тип двигуна	stardrive_engine_type	int	FK(stardrive_engine_types.id), NN
Категорія двигуна	stardrive_engine_category	int	FK(stardrive_engine_categories.id), NN

Таблиця – Структура таблиці “stasis_webs”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Кількість екіпажу	workspaces	int	NN
Вартість	cost	int	U, NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “stasis_webs_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Система стазису	stasis_web	int	FK(stasis_webs.id), NN

Таблиця – Структура таблиці “defensive_ecms”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Кількість екіпажу	workspaces	int	NN
Вартість	cost	int	U, NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “defensive_ecms_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Система електронних контрзаходів	defensive_ecm	int	FK(defensive_ecms.id), NN

Таблиця – Структура таблиці “control_rooms”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Потужність комп’ютерів	base_complexity	int	NN
Рівень сенсорних систем	comm_sensor_level	int	U, NN
Максимальна кількість контрольних інтерфейсів	max_control_stations	int	NN
Вартість контрольного інтерфейсу	control_station_cost	int	NN
Кількість екіпажу	workspaces	int	NN
Вартість	cost	int	U, NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “control_rooms_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Контрольна система	control_room	int	FK(control_rooms.id), NN
Кількість контрольних інтерфейсів	control_stations	int	NN

Таблиця – Структура таблиці “contragravity_lifters”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Кількість екіпажу	workspaces	int	NN
Вартість	cost	int	U, NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “contragravity_lifters_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Контргравітаційна система	contragravity_lifter	int	FK(contragravity_lifters.id), NN

Таблиця – Структура таблиці “sensor_array_types”

Найменування	Ідентифікатор	Тип поля	Примітки
Код типу виробничої системи	id	int	PK, NN
Назва	name	varchar(12)	U, NN
Модифікатор вартості	cost_multiplier	int	U, NN

Таблиця – Структура таблиці “sensor_arrays”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Рівень системи	array_level	int	U, NN
Кількість екіпажу	workspaces	int	NN
Вартість	cost	int	U, NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “sensor_arrays_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Сенсорна система	sensor_array	int	FK(sensor_arrays.id), NN
Тип системи	sensor_array_type	int	FK(sensor_array_types.id), NN

Таблиця – Структура таблиці “cloaking_devices_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Маскувальна система	cloaking_device	int	FK(cloaking_devices.id), NN

Таблиця – Структура таблиці “cloaking_devices”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Кількість екіпажу	workspaces	int	NN
Вартість	cost	int	U, NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “cargo_holds”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Тоннаж	capacity	float	NN, U
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “cargo_holds_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Трюм	cargo_hold	int	FK(cargo_holds.id), NN
Охолоджуваний вантаж	refrigerated_cargo	int	NN
Захищений вантаж	shielded_cargo	int	NN

Таблиця – Структура таблиці “ramscopes”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Рівень системи	array_level	int	U, NN
Кількість екіпажу	workspaces	int	NN
Вартість	cost	int	U, NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “ramscopes_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Сенсорна система	ramscope	int	FK(ramscopes.id), NN

Таблиця – Структура таблиці “power_plant_types”

Найменування	Ідентифікатор	Тип поля	Примітки
Код типу	id	int	PK, NN
Назва	name	varchar(29)	U, NN
Потужність енергії	power_points	int	NN
Можливість зниження потужності	deratable	bit	NN
Одиниця виміру часу роботи	endurance_unit	varchar(1)	NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “power_plant_variants”

Найменування	Ідентифікатор	Тип поля	Примітки
Код варіації	id	int	PK, NN
Тип силових установок	power_plant_type	int	FK(power_plant_types.id), NN
Час роботи	endurance	int	NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN

Таблиця – Структура таблиці “power_plants”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Тип силових установок	power_plant_type	int	FK(power_plant_types.id), NN
Розмір	sm	int	FK(sm.id), NN
Кількість екіпажу	workspaces	int	NN
Вартість	cost	int	U, NN

Таблиця – Структура таблиці “power_plants_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Силова установка	power_plant	int	FK(power_plants.id), NN
Варіація силових установок	power_plant_variant	int	FK(power_plant_variants.id), NN
Зменшення потужності	derated	int	NN

Таблиця – Структура таблиці “habitats”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Кількість кімнат	cabins	int	U, NN
Кількість екіпажу	workspaces	int	NN
Вартість	cost	int	U, NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “life_support”

Найменування	Ідентифікатор	Тип поля	Примітки
Код типу систем	id	int	PK, NN
Назва	name	varchar(18)	U, NN
Модифікатор кількості жилих кімнат	cabin_modifier	int	NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN

Таблиця – Структура таблиці “habitats_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Жилий відсік	habitat	int	FK(habitats.id), NN
Тип системи забезпечення життєдіяльності	life_support	int	FK(life_support.id), NN

Таблиця – Структура таблиці “rooms”

Найменування	Ідентифікатор	Тип поля	Примітки
Код кімнати	id	int	PK, NN
Назва	name	varchar(28)	U, NN
Кількість кімнат	cabins	int	U, NN
Кількість місць	occupancy	int	NN
Вартість	cost	int	NN
Розмір	size	int	NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN

Таблиця – Структура таблиці “rooms_to_habitats”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Запис жилого відсіку	habitat	int	FK(habitats_to_slots.id), NN
Кімната	room	int	FK(rooms.id), NN
Кількість	amount	int	NN

Таблиця – Структура таблиці “fuel_tanks”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Розмір	sm	int	FK(sm.id), NN
Об’єм	capacity	float	U, NN
Вартість	cost	int	U, NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “fuel_tanks_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Паливний бак	fuel_tank	int	FK(fuel_tanks.id), NN
Тип палива	fuel	int	FK(fuel.id), NN

Таблиця – Структура таблиці “engines”

Найменування	Ідентифікатор	Тип поля	Примітки
Код системи	id	int	PK, NN
Тип двигуна	engine_type	int	FK(engine_types.id), NN
Розмір	sm	int	FK(sm.id), NN
Вартість	cost	int	U, NN
Кількість екіпажу	workspaces	int	NN

Таблиця – Структура таблиці “engine_variations”

Найменування	Ідентифікатор	Тип поля	Примітки
Код варіації	id	int	PK, NN
Тип двигуна	engine_type	int	FK(engine_types.id), NN
Запас палива на один бак	delta_v_per_tank	float	NN
Прискорення	acceleration	float	NN
Паливо	fuel	int	FK(fuel.id), NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN

Таблиця – Структура таблиці “fuel”

Найменування	Ідентифікатор	Тип поля	Примітки
Код палива	id	int	PK, NN
Назва	name	varchar(29)	U, NN
Вартість за тонну	cost_per_tonbig	int	NN

Таблиця – Структура таблиці “weapon_placements”

Найменування	Ідентифікатор	Тип поля	Примітки
Код виду	id	int	PK, NN
Назва	name	varchar(12)	U, NN

Таблиця – Структура таблиці “rates_of_fire”

Найменування	Ідентифікатор	Тип поля	Примітки
Код рівня	id	int	PK, NN
Назва	name	varchar(15)	U, NN
Скорострільність	rate_of_fire	int	U, NN

Таблиця – Структура таблиці “weapon_types”

Найменування	Ідентифікатор	Тип поля	Примітки
Код типу	id	int	PK, NN
Назва	name	varchar(7)	U, NN

Таблиця – Структура таблиці “engine_types”

Найменування	Ідентифікатор	Тип поля	Примітки
Код типу	id	int	PK, NN
Назва	name	varchar(29)	U, NN
Використання енергії	power_draw	int	NN
Наявність варіанта з підвищеною потужністю	high_thrust_modifier	bit	NN
Можливість необмеженого польоту в атмосфері	ram_rocket	bit	NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “engines_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Двигун	engine	int	FK(engines.id), NN
Варіація двигуна	engine_variation	int	FK(engine_variations.id), NN
Модель з високою потужністю	high_thrust	bit	NN
Модель, яка не вимагає палива в атмосфері	ram_rocket	bit	NN

Таблиця – Структура таблиці “weapon_power”

Найменування	Ідентифікатор	Тип поля	Примітки
Код потужності	id	int	PK, NN
Потужність в кілоджоулях	power_output	bigint	U, NN
Пошкодження першого рівня	damage1	varchar(8)	U, NN
Пошкодження другого рівня	damage2	varchar(8)	U, NN

Таблиця – Структура таблиці “weapon_types”

Найменування	Ідентифікатор	Тип поля	Примітки
Код типу	id	int	PK, NN
Назва	name	varchar(29)	U, NN
Тип озброєння	weapon_type	int	FK(weapon_types.id), NN
Використання енергії	power_draw	int	NN
Рівень пошкоджень	damage_level	int	NN
Пробиття броні	armor_divisor	int	NN
Типи пошкоджень	damage_types	varchar(30)	NN
Мінімальний технологічний рівень	min_tl	int	FK(tl.id), NN

Таблиця – Структура таблиці “weapon_system_types”

Найменування	Ідентифікатор	Тип поля	Примітки
Код типу	id	int	PK, NN
Назва	name	varchar(9)	U, NN
Розміщення	placement	int	FK(hull_sections.id), NN

Таблиця – Структура таблиці “weapon_calibers”

Найменування	Ідентифікатор	Тип поля	Примітки
Код калібру	id	int	PK, NN
Калібр в сантиметрах	caliber	float	U, NN
Пошкодження	damage	varchar(6)	U, NN
Вага снаряду	weight_per_shot	float	U, NN

Таблиця – Структура таблиці “beam_ranges”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Підтип озброєння	subtype	int	FK(weapon_subtypes.id), NN
Потужність	beam_power	int	FK(weapon_power.id), NN
Дистанція повної потужності	half_range	int	U, NN
Максимальна дистанція	max_range	int	U, NN

Таблиця – Структура таблиці “weapon_systems_to_slots”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Слот	slot	int	FK(hull_slots.id), NN
Система озброєння	weapon_system	int	FK(weapon_systems_to_slots.id), NN

Таблиця – Структура таблиці “weapon_system_configurations”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Тип системи озброєння	weapon_system_type	int	FK(weapon_system_types.id), NN
Кількість озброєння	weapon_count	int	NN
Розмір	sm	int	FK(sm.id), NN
Стандартний калібр	default_caliber	int	FK(weapon_calibers.id), NN
Стандартна потужність	default_power	int	FK(weapon_power.id), NN
Стандартна кількість снарядів	shots_per_weapon	int	NN
Вага одиниці озброєння в тоннах	weight_per_weapon	int	NN
Вартість	cost	int	NN
Кількість екіпажу	workspaces	int	NN

Таблиця – Структура таблиці “weapon_systems”

Найменування	Ідентифікатор	Тип поля	Примітки
Код запису	id	int	PK, NN
Підтип системи озброєння	subtype	int	FK(weapon_subtypes.id), NN
Розміщення озброєння	weapon_placement	int	FK(weapon_placements.id), NN
Скорострільність	rate_of_fire	int	FK(rates_of_fire.id), NN
Кількість встановлених одиниць озброєння	installed_weapons	int	NN
Конфігурація	configuration	int	FK(weapon_system_configurations.id), NN

ДОДАТОК В. Сценарій генерації та заповнення БД

```

CREATE DATABASE ccdb
GO
USE ccdb
--star systems
CREATE TABLE star_systems(
    id          INT PRIMARY KEY IDENTITY(1,1),
    name        VARCHAR(20) NOT NULL UNIQUE,
    star_mass   FLOAT NOT NULL,
)
GO
CREATE TABLE stellar_object_types(
    id          INT PRIMARY KEY IDENTITY(1, 1),
    name        VARCHAR(17) NOT NULL UNIQUE
)
GO
CREATE TABLE stellar_objects(
    id          INT PRIMARY KEY IDENTITY(1, 1),
    star_system INT FOREIGN KEY REFERENCES
star_systems(id),
    stellar_object_type INT NOT NULL REFERENCES
stellar_object_types(id),
    name        VARCHAR(20) NOT NULL UNIQUE,
    orbital_radius  FLOAT NOT NULL,
    orbital_eccentricity  FLOAT NOT NULL,
    orbital_angle   INT NOT NULL,
    mass            FLOAT NOT NULL,
    planetary_radius  FLOAT NOT NULL,
    gravity         FLOAT NOT NULL,
    object_description  VARCHAR(500) NOT NULL
)
GO
--spaceships
CREATE TABLE tl(
    id          INT PRIMARY KEY,
    name        VARCHAR(21) NOT NULL UNIQUE,
    computer_complexity INT NOT NULL
)
GO
CREATE TABLE sm(
    id          INT PRIMARY KEY,
    spaceship_mass  INT NOT NULL UNIQUE,
    spaceship_length INT NOT NULL UNIQUE,
    dhp           INT NOT NULL UNIQUE,

```

```

        handling                INT NOT NULL,
        stability                INT NOT NULL,
        stealth_cost             INT NOT NULL UNIQUE,
        dynamic_chameleon_cost   INT NOT NULL UNIQUE
    )
GO
CREATE TABLE automation_levels(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    name              VARCHAR(20) NOT NULL UNIQUE,
    crew_factor       FLOAT NOT NULL UNIQUE,
    cost_per_crew      INT NOT NULL,
    min_sm            INT NOT NULL REFERENCES
sm(id),
    min_tl            INT NOT NULL REFERENCES
tl(id)
)
GO
CREATE TABLE wings(
    id                INT PRIMARY KEY IDENTITY(1, 1),      --
stops at SM +12
    sm                INT NOT NULL UNIQUE REFERENCES sm(id),
    cost              INT NOT NULL UNIQUE
)
GO
CREATE TABLE spin_gravity(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    --starts at SM +8
    sm                INT NOT NULL REFERENCES sm(id),
    max_g             FLOAT NOT NULL UNIQUE,
    cost              INT NOT NULL UNIQUE
)
GO
CREATE TABLE ship_conditions(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    name              VARCHAR(10) NOT NULL UNIQUE,
    price_modifier    FLOAT NOT NULL UNIQUE,
    health_modifier    INT NOT NULL UNIQUE
)
CREATE TABLE spaceships(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    name              VARCHAR(50) NOT NULL UNIQUE,
    tl                INT NOT NULL REFERENCES tl(id),
    sm                INT NOT NULL REFERENCES sm(id),
    streamlined        BIT NOT NULL,
    winged             BIT NOT NULL,
    spin_gravity       BIT NOT NULL,

```

```

        stealth          BIT NOT NULL,
        dynamic_chameleon BIT NOT NULL,
        automation        INT NOT NULL REFERENCES
automation_levels(id),
        condition        INT NOT NULL REFERENCES
ship_conditions(id)
    )
GO
CREATE TABLE hull_sections(
    id          INT PRIMARY KEY IDENTITY(1, 1),
    name VARCHAR(7) NOT NULL UNIQUE
)
GO
CREATE TABLE hull_locations(
    id          INT PRIMARY KEY IDENTITY(1, 1),
    name VARCHAR(4) NOT NULL UNIQUE
)
GO
CREATE TABLE hull_slots(
    id          INT PRIMARY KEY IDENTITY(1, 1),
    ship        INT NOT NULL REFERENCES spaceships(id),
    section     INT NOT NULL REFERENCES hull_sections(id),
    location    INT NOT NULL REFERENCES hull_locations(id)
)
GO
CREATE TABLE engine_rooms(
    id          INT PRIMARY KEY IDENTITY(1, 1),
    sm          INT NOT NULL REFERENCES sm(id),
    workspaces  INT NOT NULL,
    cost        INT NOT NULL UNIQUE,
    min_t1      INT NOT NULL REFERENCES t1(id),
    placement   INT NOT NULL REFERENCES hull_sections(id)
)
GO
CREATE TABLE engine_rooms_to_slots(
    id          INT PRIMARY KEY IDENTITY(1, 1),
    slot        INT NOT NULL REFERENCES hull_slots(id),
    engine_room INT NOT NULL REFERENCES engine_rooms(id)
)
GO
CREATE TABLE external_clamps(
    id          INT PRIMARY KEY IDENTITY(1, 1),
    sm          INT NOT NULL REFERENCES sm(id),
    cost        INT NOT NULL UNIQUE,
    min_t1      INT NOT NULL REFERENCES t1(id),
    placement   INT NOT NULL REFERENCES hull_sections(id)

```

```

)
CREATE TABLE external_clamps_to_slots(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    slot              INT NOT NULL REFERENCES hull_slots(id),
    external_clamp    INT NOT NULL REFERENCES external_clamps(id)
)
GO
CREATE TABLE factories(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    sm                INT NOT NULL REFERENCES sm(id),
    output_base       INT NOT NULL UNIQUE,
    workspaces        INT NOT NULL,
    cost              INT NOT NULL UNIQUE,
    placement         INT NOT NULL REFERENCES hull_sections(id)
)
GO
CREATE TABLE factory_types(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    name              VARCHAR(11) NOT NULL UNIQUE,
    output_multiplier  FLOAT NOT NULL,
    output_unit        VARCHAR(4) NOT NULL,
    cost_multiplier    INT NOT NULL UNIQUE,
    min_t1            INT NOT NULL REFERENCES t1(id)
)
GO
CREATE TABLE factories_to_slots(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    slot              INT NOT NULL REFERENCES hull_slots(id),
    factory           INT NOT NULL REFERENCES factories(id),
    factory_type      INT NOT NULL REFERENCES factory_types(id)
)
GO
CREATE TABLE force_screen_types(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    name              VARCHAR(11) NOT NULL UNIQUE,
    power_draw        INT NOT NULL UNIQUE
)
GO
CREATE TABLE force_screens(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    force_screen_type INT NOT NULL REFERENCES
force_screen_types(id),
    sm                INT NOT NULL REFERENCES sm(id),
    dr                INT NOT NULL,
    workspaces        INT NOT NULL,
    cost              INT NOT NULL,

```

[illegible]

```

        slot                                INT NOT NULL REFERENCES
hull_slots(id),
        mining_refinery_system              INT NOT NULL REFERENCES
mining_refinery_systems(id)
)
GO
CREATE TABLE open_spaces(
    id                                INT PRIMARY KEY IDENTITY(1, 1),
    sm                                INT NOT NULL REFERENCES sm(id),
    areas                             INT NOT NULL UNIQUE,
    workspaces                         INT NOT NULL,
    cost                              INT NOT NULL UNIQUE,
    min_tl                             INT NOT NULL REFERENCES tl(id),
    placement                          INT NOT NULL REFERENCES hull_sections(id)
)
GO
CREATE TABLE open_spaces_to_slots(
    id                                INT PRIMARY KEY IDENTITY(1, 1),
    slot                              INT NOT NULL REFERENCES hull_slots(id),
    open_space                         INT NOT NULL REFERENCES open_spaces(id)
)
GO
CREATE TABLE robot_arms(
    id                                INT PRIMARY KEY IDENTITY(1, 1),
    sm                                INT NOT NULL REFERENCES sm(id),
    workspaces                         INT NOT NULL,
    cost                              INT NOT NULL UNIQUE,
    min_tl                             INT NOT NULL REFERENCES tl(id),
    placement                          INT NOT NULL REFERENCES hull_sections(id)
)
GO
CREATE TABLE robot_arms_to_slots(
    id                                INT PRIMARY KEY IDENTITY(1, 1),
    slot                              INT NOT NULL REFERENCES hull_slots(id),
    robot_arm                          INT NOT NULL REFERENCES robot_arms(id),
    grabber                            BIT NOT NULL
)
GO
CREATE TABLE soft_landing_systems(
    id                                INT PRIMARY KEY IDENTITY(1, 1),
    sm                                INT NOT NULL REFERENCES sm(id),
    cost                              INT NOT NULL UNIQUE,
    min_tl                             INT NOT NULL REFERENCES tl(id),
    placement                          INT NOT NULL REFERENCES hull_sections(id)
)
GO

```

```

CREATE TABLE soft_landing_systems_to_slots(
    id                      INT PRIMARY KEY IDENTITY(1, 1),
    slot                    INT NOT NULL REFERENCES hull_slots(id),
    soft_landing_system     INT NOT NULL REFERENCES
soft_landing_systems(id)
)
GO
CREATE TABLE stardrive_engines(
    id                      INT PRIMARY KEY IDENTITY(1, 1),
    sm                      INT NOT NULL REFERENCES sm(id),
    workspaces              INT NOT NULL,
    cost                    INT NOT NULL UNIQUE,
    placement               INT NOT NULL REFERENCES hull_sections(id)
)
GO
CREATE TABLE stardrive_engine_types(
    id                      INT PRIMARY KEY IDENTITY(1, 1),
    name                    VARCHAR(22) NOT NULL UNIQUE,
    ftl_speed               INT NOT NULL,
    power_draw              INT NOT NULL,
    cost_multiplier         INT NOT NULL
)
GO
CREATE TABLE stardrive_engine_categories(
    id                      INT PRIMARY KEY IDENTITY(1, 1),
    name                    VARCHAR(17) NOT NULL UNIQUE
)
GO
CREATE TABLE stardrive_engines_to_slots(
    id                      INT PRIMARY KEY
IDENTITY(1, 1),
    slot                    INT NOT NULL REFERENCES
hull_slots(id),
    stardrive_engine        INT NOT NULL REFERENCES
stardrive_engines(id),
    stardrive_engine_type   INT NOT NULL REFERENCES
stardrive_engine_types(id),
    stardrive_engine_category INT NOT NULL REFERENCES
stardrive_engine_categories(id)
)
GO
CREATE TABLE stasis_webs(
    id                      INT PRIMARY KEY IDENTITY(1, 1),
    sm                      INT NOT NULL REFERENCES sm(id),
    workspaces              INT NOT NULL,
    cost                    INT NOT NULL UNIQUE,

```

```

        min_tl          INT NOT NULL REFERENCES tl(id),
        placement       INT NOT NULL REFERENCES hull_sections(id)
    )
GO
CREATE TABLE stasis_webs_to_slots(
    id                  INT PRIMARY KEY IDENTITY(1, 1),
    slot               INT NOT NULL REFERENCES hull_slots(id),
    stasis_web         INT NOT NULL REFERENCES stasis_webs(id)
)
GO
CREATE TABLE defensive_ecms(
    id                  INT PRIMARY KEY IDENTITY(1, 1),
    sm                 INT NOT NULL REFERENCES sm(id),
    workspaces         INT NOT NULL,
    cost               INT NOT NULL UNIQUE,
    min_tl             INT NOT NULL REFERENCES tl(id),
    placement          INT NOT NULL REFERENCES hull_sections(id)
)
GO
CREATE TABLE defensive_ecms_to_slots(
    id                  INT PRIMARY KEY IDENTITY(1, 1),
    slot               INT NOT NULL REFERENCES hull_slots(id),
    defensive_ecm      INT NOT NULL REFERENCES
defensive_ecms(id)
)
GO
CREATE TABLE control_rooms(
    id                  INT PRIMARY KEY IDENTITY(1, 1),
    sm                 INT NOT NULL REFERENCES sm(id),
    base_complexity    INT NOT NULL,
    comm_sensor_level  INT NOT NULL UNIQUE,
    max_control_stations INT NOT NULL,
    control_station_cost INT NOT NULL,
    workspaces         INT NOT NULL,
    cost               INT NOT NULL UNIQUE,
    min_tl             INT NOT NULL REFERENCES tl(id),
    placement          INT NOT NULL REFERENCES
hull_sections(id)
)
GO
CREATE TABLE control_rooms_to_slots(
    id                  INT PRIMARY KEY IDENTITY(1, 1),
    slot               INT NOT NULL REFERENCES hull_slots(id),
    control_room       INT NOT NULL REFERENCES
control_rooms(id),
    control_stations   INT NOT NULL

```

```

)
GO
CREATE TABLE contragravity_lifters(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    sm                INT NOT NULL REFERENCES sm(id),
    workspaces        INT NOT NULL,
    cost              INT NOT NULL UNIQUE,
    placement          INT NOT NULL REFERENCES hull_sections(id)
)
GO
CREATE TABLE contragravity_lifters_to_slots(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    slot              INT NOT NULL REFERENCES
hull_slots(id),
    contragravity_lifter INT NOT NULL REFERENCES
contragravity_lifters(id)
)
GO
CREATE TABLE cloaking_devices(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    sm                INT NOT NULL REFERENCES sm(id),
    workspaces        INT NOT NULL,
    cost              INT NOT NULL UNIQUE,
    placement          INT NOT NULL REFERENCES hull_sections(id)
)
GO
CREATE TABLE cloaking_devices_to_slots(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    slot              INT NOT NULL REFERENCES hull_slots(id),
    cloaking_device   INT NOT NULL REFERENCES
cloaking_devices(id)
)
GO
CREATE TABLE habitats(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    sm                INT NOT NULL REFERENCES sm(id),
    cabins            INT NOT NULL UNIQUE,
    workspaces        INT NOT NULL,
    cost              INT NOT NULL UNIQUE,
    placement          INT NOT NULL REFERENCES hull_sections(id)
)
GO
CREATE TABLE life_support(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    name              VARCHAR(18) NOT NULL UNIQUE,
    cabin_modifier    INT NOT NULL UNIQUE,

```

```

        min_tl          INT NOT NULL REFERENCES tl(id)
    )
GO
CREATE TABLE habitats_to_slots(
    id                  INT PRIMARY KEY IDENTITY(1, 1),
    slot                INT NOT NULL REFERENCES hull_slots(id),
    habitat             INT NOT NULL REFERENCES habitats(id),
    life_support        INT NOT NULL REFERENCES life_support(id)
)
GO
CREATE TABLE rooms(
    id                  INT PRIMARY KEY IDENTITY(1, 1),
    name                VARCHAR(28) NOT NULL UNIQUE,
    occupancy           INT NOT NULL,
    cost                INT NOT NULL,
    size                FLOAT NOT NULL,
    min_tl              INT NOT NULL REFERENCES tl(id)
)
GO
CREATE TABLE rooms_to_habitats(
    id                  INT PRIMARY KEY IDENTITY(1, 1),
    habitat             INT NOT NULL REFERENCES habitats_to_slots(id),
    room                INT NOT NULL REFERENCES rooms(id),
    amount              INT NOT NULL
)
GO
CREATE TABLE fuel_tanks(
    id                  INT PRIMARY KEY IDENTITY(1, 1),
    sm                  INT NOT NULL REFERENCES sm(id),
    capacity            FLOAT NOT NULL UNIQUE,
    cost                INT NOT NULL UNIQUE,
    min_tl              INT NOT NULL REFERENCES tl(id),
    placement           INT NOT NULL REFERENCES hull_sections(id)
)
GO
CREATE TABLE fuel(
    id                  INT PRIMARY KEY IDENTITY(0, 1),
    name                VARCHAR(29) NOT NULL UNIQUE,
    cost_per_ton        BIGINT NOT NULL --not in thouthands
)
GO
CREATE TABLE fuel_tanks_to_slots(
    id                  INT PRIMARY KEY IDENTITY(1, 1),
    slot                INT NOT NULL REFERENCES hull_slots(id),
    fuel_tank           INT NOT NULL REFERENCES fuel_tanks(id),
    fuel                INT NOT NULL REFERENCES fuel(id)

```

```

)
GO
CREATE TABLE engine_types(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    name              VARCHAR(29) NOT NULL, --null means
reactionless
    power_draw        INT NOT NULL,
    high_thrust_modifier INT NOT NULL, --0 means high
thrust option isn't available,
    ram_rocket        BIT NOT NULL, --1 means that it can be a ram
rocket
    placement         INT NOT NULL REFERENCES hull_sections(id)
)
GO
CREATE TABLE engine_variations(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    engine_type        INT NOT NULL REFERENCES
engine_types(id),
    delta_v_per_tank   FLOAT NOT NULL, --0 for jet engines
acceleration         FLOAT NOT NULL,
    fuel               INT REFERENCES fuel(id), --null means
reactionless
    min_t1             INT NOT NULL REFERENCES t1(id)
)
GO
CREATE TABLE engines(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    engine_type        INT NOT NULL REFERENCES
engine_types(id),
    sm                 INT NOT NULL REFERENCES sm(id),
    cost               INT NOT NULL,
    workspaces         INT NOT NULL
)
GO
CREATE TABLE engines_to_slots(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    slot               INT NOT NULL REFERENCES hull_slots(id),
    engine             INT NOT NULL REFERENCES engines(id),
    engine_variation   INT NOT NULL REFERENCES
engine_variations(id),
    high_thrust        BIT NOT NULL,
    ram_rocket         BIT NOT NULL
)
GO
CREATE TABLE armor_types(
    id                INT PRIMARY KEY IDENTITY(1, 1),

```

```

        name                VARCHAR(26) NOT NULL UNIQUE,
        min_tl              INT NOT NULL REFERENCES tl(id)
    )
GO
CREATE TABLE armor_systems(
    id                      INT PRIMARY KEY IDENTITY(1, 1),
    sm                     INT NOT NULL REFERENCES sm(id),
    armor_type             INT NOT NULL REFERENCES
armor_types(id),
    streamlined_dr        INT NOT NULL,
    unstreamlined_dr      INT NOT NULL,
    cost                  INT NOT NULL,
    placement             INT NOT NULL REFERENCES
hull_sections(id)
)
GO
CREATE TABLE armor_to_slots(
    id                    INT PRIMARY KEY IDENTITY(1, 1),
    slot                 INT NOT NULL REFERENCES hull_slots(id),
    armor_system         INT NOT NULL REFERENCES armor_systems(id),
    hardening           INT NOT NULL
)
GO
CREATE TABLE hangar_bays(
    id                  INT PRIMARY KEY IDENTITY(1, 1),
    sm                 INT NOT NULL REFERENCES sm(id),
    capacity           INT NOT NULL UNIQUE,
    launch_rate        INT NOT NULL UNIQUE,
    workspaces         INT NOT NULL,
    cost              INT NOT NULL UNIQUE,
    min_tl            INT NOT NULL REFERENCES tl(id),
    placement         INT NOT NULL REFERENCES
hull_sections(id)
)
GO
CREATE TABLE hangar_bays_to_slots(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    slot             INT NOT NULL REFERENCES hull_slots(id),
    hangar_bay       INT NOT NULL REFERENCES hangar_bays(id),
    optimized        INT REFERENCES sm(id)
)
GO
CREATE TABLE sensor_arrays(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    sm               INT NOT NULL REFERENCES sm(id),
    array_level      INT NOT NULL UNIQUE,

```

```

        workspaces                INT NOT NULL,
        cost                      INT NOT NULL UNIQUE,
        min_tl                   INT NOT NULL REFERENCES tl(id),
        placement                INT NOT NULL REFERENCES
hull_sections(id)
    )
GO
CREATE TABLE sensor_array_types(
    id                          INT PRIMARY KEY IDENTITY(1, 1),
    name                       VARCHAR(12) NOT NULL UNIQUE,
    cost_multiplier            INT NOT NULL
)
GO
CREATE TABLE sensor_arrays_to_slots(
    id                        INT PRIMARY KEY IDENTITY(1, 1),
    slot                    INT NOT NULL REFERENCES hull_slots(id),
    sensor_array            INT NOT NULL REFERENCES sensor_arrays(id),
    sensor_array_type       INT NOT NULL REFERENCES
sensor_array_types(id)
)
GO
CREATE TABLE cargo_holds(
    id                      INT PRIMARY KEY IDENTITY(1, 1),
    sm                    INT NOT NULL REFERENCES sm(id),
    capacity              FLOAT NOT NULL UNIQUE,
    placement            INT NOT NULL REFERENCES
hull_sections(id)
)
GO
CREATE TABLE cargo_holds_to_slots(
    id                      INT PRIMARY KEY IDENTITY(1, 1),
    slot                  INT NOT NULL REFERENCES hull_slots(id),
    cargo_hold            INT NOT NULL REFERENCES cargo_holds(id),
    refrigerated_cargo    INT NOT NULL,
    shielded_cargo        INT NOT NULL
)
GO
CREATE TABLE ramscoops(
    id                      INT PRIMARY KEY IDENTITY(1, 1),
    sm                    INT NOT NULL REFERENCES sm(id),
    workspaces            INT NOT NULL,
    cost                  INT NOT NULL UNIQUE,
    min_tl               INT NOT NULL REFERENCES tl(id),
    placement            INT NOT NULL REFERENCES
hull_sections(id)
)

```

```

GO
CREATE TABLE ramscoops_to_slots(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    slot              INT NOT NULL REFERENCES hull_slots(id),
    ramscoop           INT NOT NULL REFERENCES ramscoops(id)
)
GO
CREATE TABLE power_plant_types(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    name              VARCHAR(24) NOT NULL UNIQUE,
    power_points      INT NOT NULL,
    deratable         BIT NOT NULL,
    endurance_unit    VARCHAR(1) NOT NULL,
    placement         INT NOT NULL REFERENCES
hull_sections(id)
)
GO
CREATE TABLE power_plant_variants(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    power_plant_type  INT NOT NULL REFERENCES
power_plant_types(id),
    endurance         INT NOT NULL,
    min_t1            INT NOT NULL REFERENCES t1(id)
)
GO
CREATE TABLE power_plants(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    power_plant_type  INT NOT NULL REFERENCES
power_plant_types(id),
    sm                INT NOT NULL REFERENCES sm(id),
    workspaces        INT NOT NULL,
    cost              INT NOT NULL
)
GO
CREATE TABLE power_plants_to_slots(
    id                INT PRIMARY KEY IDENTITY(1, 1),
    slot              INT NOT NULL REFERENCES hull_slots(id),
    power_plant       INT NOT NULL REFERENCES
power_plants(id),
    power_plant_variant INT NOT NULL REFERENCES
power_plant_variants(id),
    derated           INT NOT NULL
)
GO
CREATE TABLE passenger_seating(
    id                INT PRIMARY KEY IDENTITY(1, 1),

```

```

        sm                INT NOT NULL REFERENCES sm(id),
        seats             INT NOT NULL UNIQUE,
        cost              INT NOT NULL UNIQUE,
        placement         INT NOT NULL REFERENCES hull_sections(id)
    )
GO
CREATE TABLE passenger_seating_to_slots(
    id                    INT PRIMARY KEY IDENTITY(1, 1),
    slot                 INT NOT NULL REFERENCES hull_slots(id),
    passenger_seating    INT NOT NULL REFERENCES
passenger_seating(id),
    stretchers          INT NOT NULL
)
GO
CREATE TABLE weapon_types(
    id                   INT PRIMARY KEY IDENTITY(1, 1),
    name                 VARCHAR(7) NOT NULL UNIQUE
)
GO
CREATE TABLE weapon_subtypes(
    id                   INT PRIMARY KEY IDENTITY(1, 1),
    name                 VARCHAR(30) NOT NULL UNIQUE,
    weapon_type          INT NOT NULL REFERENCES
weapon_types(id),
    power_draw           INT NOT NULL,
    damage_level         INT NOT NULL, --dDam1 or dDam2
    armor_divisor        INT NOT NULL, --0 means infinite
    damage_types         VARCHAR(30) NOT NULL,
    min_t1               INT NOT NULL REFERENCES t1(id)
)
GO
CREATE TABLE weapon_placements(
    id                   INT PRIMARY KEY IDENTITY(1, 1),
    name                 VARCHAR(12) NOT NULL UNIQUE
)
GO
CREATE TABLE rates_of_fire(
    id                   INT PRIMARY KEY IDENTITY(1, 1),
    name                 VARCHAR(15) NOT NULL UNIQUE,
    rate_of_fire         INT NOT NULL UNIQUE
)
GO
CREATE TABLE weapon_system_types(
    id                   INT PRIMARY KEY IDENTITY(1, 1),
    name                 VARCHAR(9) NOT NULL UNIQUE,

```

```

        placement          INT NOT NULL REFERENCES
hull_sections(id)
)
GO
CREATE TABLE weapon_calibers(
    id                      INT PRIMARY KEY IDENTITY(1, 1),
    caliber                 FLOAT NOT NULL UNIQUE, --expressed in cm
    damage                  VARCHAR(6) NOT NULL UNIQUE, --
expressed in multiples of 6d - "5" means 6dx5
    weight_per_shot        FLOAT NOT NULL UNIQUE --expressed
in fractions of a ton
)
GO
CREATE TABLE weapon_power(
    id                      INT PRIMARY KEY IDENTITY(1, 1),
    power_output            BIGINT NOT NULL UNIQUE, --expressed in kJ
    damage1                 VARCHAR(8) NOT NULL UNIQUE, --
expressed in multiples of 6d - "5" means 6dx5
    damage2                 VARCHAR(8) NOT NULL UNIQUE --
expressed in multiples of 6d - "5" means 6dx5
)
GO
CREATE TABLE beam_ranges(
    id                      INT PRIMARY KEY IDENTITY(1, 1),
    subtype                 INT NOT NULL REFERENCES weapon_subtypes(id),
    beam_power              INT NOT NULL REFERENCES weapon_power(id),
    half_range              INT NOT NULL,
    max_range               INT NOT NULL
)
GO
CREATE TABLE weapon_system_configurations(
    id                      INT PRIMARY KEY IDENTITY(1, 1),
    weapon_system_type      INT NOT NULL REFERENCES
weapon_system_types(id),
    weapon_count            INT NOT NULL,
    sm                      INT NOT NULL REFERENCES sm(id),
    default_caliber         INT NOT NULL REFERENCES
weapon_calibers(id),
    default_power           INT NOT NULL REFERENCES
weapon_power(id),
    shots_per_weapon        INT NOT NULL,
    weight_per_weapon       INT NOT NULL, --in tons
    cost                    INT NOT NULL,
    workspaces              INT NOT NULL
)
GO

```

```

CREATE TABLE weapon_systems(
    id                                INT PRIMARY KEY IDENTITY(1, 1),
    subtype                           INT NOT NULL REFERENCES
weapon_subtypes(id),
    weapon_placement                  INT NOT NULL REFERENCES
weapon_placements(id),
    rate_of_fire                      INT NOT NULL REFERENCES
rates_of_fire(id),
    installed_weapons                 INT NOT NULL,
    configuration                     INT NOT NULL REFERENCES
weapon_system_configurations(id)
)
GO
CREATE TABLE weapon_systems_to_slots(
    id                                INT PRIMARY KEY IDENTITY(1, 1),
    slot                             INT NOT NULL REFERENCES hull_slots(id),
    weapon_system                     INT NOT NULL REFERENCES
weapon_systems(id)
)
GO
INSERT INTO stellar_object_types(name)
VALUES ('Terrestrial Planet'), ('Gas Giant'), ('Asteroid Belt')
GO
INSERT INTO tl(id, name, computer_complexity)
VALUES (0, 'Stone Age', -999), (1, 'Bronze Age', -999), (2,
'Iron Age', -999), (3, 'Medieval', -999), (4, 'Age of Sail', -
999),
(5, 'Industrial Revolution', -999), (6, 'Mechanized Age', -999),
(7, 'Nuclear Age', -6), (8, 'Digital Age', -4), (9, 'Microtech
Age', -2),
(10, 'Robotic Age', 0), (11, 'Age of Exotic Matter', 1), (12,
'Age of Miracles', -2)
GO
INSERT INTO sm (id, spaceship_mass, spaceship_length, dhp,
handling, stability, stealth_cost, dynamic_chameleon_cost)
VALUES (4, 10, 10, 15, 0, 4, 125, 90), (5, 30, 15, 20, 0, 4,
250, 180), (6, 100, 20, 30, 0, 4, 500, 350),
(7, 300, 30, 50, -1, 5, 1000, 700), (8, 1000, 50, 70, -1, 5,
2000, 1500), (9, 3000, 70, 100, -1, 5, 5000, 3000),
(10, 10000, 100, 150, -2, 5, 10000, 7000), (11, 30000, 150, 200,
-2, 5, 20000, 15000), (12, 100000, 200, 300, -2, 5, 50000,
35000),
(13, 300000, 300, 500, -3, 5, 100000, 70000), (14, 1000000, 500,
700, -3, 5, 200000, 150000),
(15, 3000000, 700, 1000, -3, 5, 500000, 350000)
GO

```

```

INSERT INTO automation_levels (name, crew_factor, cost_per_crew,
min_sm, min_tl)
VALUES ('Regular Automation', 1, 0, 4, 7), ('Total Automation',
0, 5000, 4, 7), ('High Automation', 0.1, 1000, 12, 7),
('Lacks Automation', 10, 0, 4, 0)
GO
INSERT INTO wings (sm, cost)
VALUES (4, 50), (5, 150), (6, 500), (7, 1500), (8, 5000), (9,
15000), (10, 50000), (11, 150000), (12, 500000)
GO
INSERT INTO spin_gravity (sm, max_g, cost)
VALUES (8, 0.1, 100), (9, 0.15, 300), (10, 0.2, 1000), (11, 0.3,
3000), (12, 0.5, 10000), (13, 0.7, 30000), (14, 1, 1000000),
(15, 1.5, 300000)
GO
INSERT INTO ship_conditions (name, price_modifier,
health_modifier)
VALUES ('Good', 1, 0), ('Cheap', 0.5, -2), ('Very Cheap', 0.2, -
4)
GO
INSERT INTO hull_sections (name)
VALUES ('Front'), ('Center'), ('Rear'), ('Hull'), ('Any'),
('Special')
GO
INSERT INTO hull_locations (name)
VALUES ('1'), ('2'), ('3'), ('4'), ('5'), ('6'), ('Core')
GO
INSERT INTO engine_rooms (sm, workspaces, cost, min_tl,
placement)
VALUES (5, 1, 15, 7, 5), (6, 1, 30, 7, 5), (7, 1, 100, 7, 5),
(8, 1, 300, 7, 5), (9, 2, 1000, 7, 5)
GO
INSERT INTO external_clamps (sm, cost, min_tl, placement)
VALUES (4, 1, 7, 4), (5, 3, 7, 4), (6, 10, 7, 4), (7, 30, 7, 4),
(8, 100, 7, 4), (9, 300, 7, 4), (10, 1000, 7, 4), (11, 3000, 7,
4),
(12, 10000, 7, 4), (13, 30000, 7, 4), (14, 100000, 7, 4), (15,
300000, 7, 4)
GO
INSERT INTO factories (sm, output_base, workspaces, cost,
placement)
VALUES (6, 5, 0, 5000, 5), (7, 15, 0, 15000, 5), (8, 50, 0,
50000, 5), (9, 150, 0, 150000, 5), (10, 500, 1, 500000, 5),
(11, 1500, 3, 1500000, 5), (12, 5000, 10, 5000000, 5), (13,
15000, 30, 15000000, 5), (14, 50000, 100, 50000000, 5),
(15, 150000, 300, 150000000, 5)

```

```

GO
INSERT INTO factory_types (name, output_multiplier, output_unit,
cost_multiplier, min_tl)
VALUES ('Fabricator', 1, '$', 1, 8), ('Robofac', 2, '$', 2, 10),
('Nanofactory', 20, '$', 4, 11), ('Replicator', 0.1, 'lbs.', 20, 12)
GO
INSERT INTO force_screen_types(name, power_draw)
VALUES ('Light', 1), ('Heavy', 2)
GO
INSERT INTO force_screens (force_screen_type, sm, dr,
workspaces, cost, min_tl, placement)
VALUES (1, 4, 15, 0, 150, 11, 5), (2, 4, 30, 0, 500, 11, 5), (1,
4, 20, 0, 150, 12, 5), (2, 4, 40, 0, 500, 12, 5),
(1, 5, 20, 0, 500, 11, 5), (2, 5, 40, 0, 1500, 11, 5), (1, 5,
30, 0, 500, 12, 5), (2, 4, 60, 0, 1500, 12, 5),
(1, 6, 30, 0, 1500, 11, 5), (2, 6, 60, 0, 5000, 11, 5), (1, 6,
50, 0, 1500, 12, 5), (2, 6, 100, 0, 5000, 12, 5),
(1, 7, 50, 0, 5000, 11, 5), (2, 7, 100, 0, 15000, 11, 5), (1, 7,
75, 0, 5000, 12, 5), (2, 7, 150, 0, 15000, 12, 5),
(1, 8, 75, 0, 15000, 11, 5), (2, 8, 150, 0, 50000, 11, 5), (1,
8, 100, 0, 15000, 12, 5), (2, 8, 200, 0, 50000, 12, 5),
(1, 9, 100, 0, 50000, 11, 5), (2, 9, 200, 0, 150000, 11, 5), (1,
9, 150, 0, 50000, 12, 5), (2, 9, 300, 0, 150000, 12, 5),
(1, 10, 150, 0, 150000, 11, 5), (2, 10, 300, 0, 500000, 11, 5),
(1, 10, 250, 0, 150000, 12, 5), (2, 10, 500, 0, 500000, 12, 5),
(1, 11, 250, 0, 500000, 11, 5), (2, 11, 500, 0, 1500000, 11, 5),
(1, 11, 350, 0, 500000, 12, 5), (2, 11, 700, 0, 1500000, 12, 5),
(1, 12, 350, 0, 1500000, 11, 5), (2, 12, 700, 0, 5000000, 11,
5), (1, 12, 500, 0, 1500000, 12, 5), (2, 12, 1000, 0, 5000000,
12, 5),
(1, 13, 500, 0, 5000000, 11, 5), (2, 13, 1000, 0, 15000000, 11,
5), (1, 13, 750, 0, 5000000, 12, 5), (2, 13, 1500, 0, 15000000,
12, 5),
(1, 14, 750, 0, 15000000, 11, 5), (2, 14, 1500, 0, 50000000, 11,
5), (1, 14, 1000, 0, 15000000, 12, 5), (2, 14, 2000, 0,
50000000, 12, 5),
(1, 15, 1000, 0, 50000000, 11, 5), (2, 15, 2000, 0, 150000000,
11, 5), (1, 15, 1500, 0, 50000000, 12, 5), (2, 15, 3000, 0,
150000000, 12, 5)
GO
INSERT INTO jump_gates (sm, max_tonnage, workspaces, cost,
placement)
VALUES (9, 100, 1, 150000, 4), (10, 300, 1, 500000, 4), (11,
1000, 3, 1500000, 4), (12, 3000, 10, 5000000, 4),
(13, 10000, 30, 50000000, 4), (14, 30000, 100, 150000000, 4),
(15, 100000, 300, 500000000, 4)

```

```

GO
INSERT INTO mining_refinery_types
VALUES ('Mining'), ('Refinery')
GO
INSERT INTO mining_refinery_systems (sm, system_type,
resource_output, workspaces, cost, min_tl, placement)
VALUES (4, 1, 0.05, 0, 10, 7, 5), (4, 2, 0.15, 0, 10, 7, 5), (5,
1, 0.15, 0, 30, 7, 5), (5, 2, 0.5, 0, 30, 7, 5), (6, 1, 0.5, 0,
100, 7, 5), (6, 2, 1.5, 0, 100, 7, 5), (7, 1, 1.5, 0, 300, 7,
5), (7, 2, 5, 0, 300, 7, 5), (8, 1, 5, 0, 1000, 7, 5),
(8, 2, 15, 0, 1000, 7, 5), (9, 1, 15, 0, 3000, 7, 5), (9, 2, 50,
0, 3000, 7, 5), (10, 1, 50, 1, 10000, 7, 5),
(10, 2, 150, 1, 10000, 7, 5), (11, 1, 150, 3, 30000, 7, 5), (11,
2, 500, 3, 30000, 7, 5), (12, 1, 500, 10, 100000, 7, 5),
(12, 2, 1500, 10, 100000, 7, 5), (13, 1, 1500, 30, 300000, 7,
5), (13, 2, 5000, 30, 300000, 7, 5), (14, 1, 5000, 100, 1000000,
7, 5), (14, 2, 15000, 100, 1000000, 7, 5), (15, 1, 15000, 300,
3000000, 7, 5), (15, 2, 50000, 300, 3000000, 7, 5)
GO
INSERT INTO open_spaces (sm, areas, workspaces, cost, min_tl,
placement)
VALUES (8, 1, 0, 100, 7, 5), (9, 2, 0, 200, 7, 5), (10, 5, 1,
500, 7, 5), (11, 10, 3, 1000, 7, 5), (12, 20, 10, 2000, 7, 5),
(13, 50, 30, 5000, 7, 5), (14, 100, 100, 10000, 7, 5), (15, 200,
300, 20000, 7, 5)
GO
INSERT INTO robot_arms (sm, workspaces, cost, min_tl, placement)
VALUES (4, 0, 100, 8, 4), (5, 0, 300, 8, 4), (6, 0, 1000, 8, 4),
(7, 0, 3000, 8, 4), (8, 0, 10000, 8, 4), (9, 0, 30000, 8, 4),
(10, 1, 100000, 8, 4), (11, 3, 300000, 8, 4), (12, 10, 1000000,
8, 4), (13, 30, 3000000, 8, 4), (14, 100, 10000000, 8, 4),
(15, 300, 30000000, 8, 4)
GO
INSERT INTO soft_landing_systems (sm, cost, min_tl, placement)
VALUES (4, 10, 7, 4), (5, 50, 7, 4), (6, 100, 7, 4), (7, 200, 7,
4), (8, 500, 7, 4), (9, 1000, 7, 4), (10, 2000, 7, 4),
(11, 5000, 7, 4), (12, 10000, 7, 4), (13, 20000, 7, 4), (14,
50000, 7, 4), (15, 100000, 7, 4)
GO
INSERT INTO stardrive_engines (sm, workspaces, cost, placement)
VALUES (4, 0, 100, 5), (5, 0, 300, 5), (6, 0, 1000, 5), (7, 0,
3000, 5), (8, 0, 10000, 5), (9, 0, 30000, 5), (10, 1, 100000, 5),
(11, 3, 300000, 5), (12, 10, 1000000, 5), (13, 30, 3000000, 5),
(14, 100, 10000000, 5), (15, 300, 30000000, 5)
GO

```

```

INSERT INTO stardrive_engine_types(name, ftl_speed, power_draw,
cost_multiplier)
VALUES ('Stardrive Engine', 1, 1, 1), ('Super Stardrive Engine',
2, 2, 5)
GO
INSERT INTO stardrive_engine_categories(name)
VALUES ('Hyperdrive'), ('Jump Drive'), ('Hyper Jump Drive'),
('Probability Drive'), ('Warp Drive')
GO
INSERT INTO stasis_webs (sm, workspaces, cost, min_tl,
placement)
VALUES (4, 0, 600, 12, 5), (5, 0, 3000, 12, 5), (6, 0, 6000, 12,
5), (7, 0, 12000, 12, 5), (8, 0, 25000, 12, 5), (9, 0, 50000,
12, 5), (10, 1, 100000, 12, 5), (11, 3, 200000, 12, 5), (12, 10,
500000, 12, 5), (13, 30, 1000000, 12, 5), (14, 100, 2000000, 12,
5), (15, 300, 5000000, 12, 5)
GO
INSERT INTO defensive_ecms (sm, workspaces, cost, min_tl,
placement)
VALUES (4, 0, 100, 7, 5), (5, 0, 300, 7, 5), (6, 0, 1000, 7, 5),
(7, 0, 3000, 7, 5), (8, 0, 10000, 7, 5), (9, 0, 30000, 7, 5),
(10, 1, 100000, 7, 5), (11, 3, 300000, 7, 5), (12, 10, 1000000,
7, 5), (13, 30, 3000000, 7, 5), (14, 100, 10000000, 7, 5),
(15, 300, 30000000, 7, 5)
GO
INSERT INTO control_rooms (sm, base_complexity,
comm_sensor_level, max_control_stations, control_station_cost,
workspaces, cost, min_tl, placement)
VALUES (4, 6, -7, 1, 10, 0, 20, 7, 5), (5, 6, -6, 1, 50, 0, 60,
7, 5), (6, 7, -5, 2, 50, 0, 200, 7, 5), (7, 7, -4, 3, 50, 0,
600, 7, 5), (8, 8, -3, 4, 50, 0, 2000, 7, 5), (9, 8, -2, 6, 50,
0, 6000, 7, 5), (10, 9, -1, 10, 50, 1, 20000, 7, 5),
(11, 9, 0, 15, 50, 3, 60000, 7, 5), (12, 10, 1, 20, 50, 10,
200000, 7, 5), (13, 10, 2, 30, 50, 30, 600000, 7, 5),
(14, 11, 3, 40, 50, 100, 2000000, 7, 5), (15, 11, 4, 60, 50,
300, 6000000, 7, 5)
GO
INSERT INTO contragravity_lifters (sm, workspaces, cost,
placement)
VALUES (4, 0, 100, 5), (5, 0, 300, 5), (6, 0, 1000, 5), (7, 0,
3000, 5), (8, 0, 10000, 5), (9, 0, 30000, 5), (10, 1, 100000, 5),
(11, 3, 300000, 5), (12, 10, 1000000, 5), (13, 30, 3000000, 5),
(14, 100, 10000000, 5), (15, 300, 30000000, 5)
GO
INSERT INTO cloaking_devices (sm, workspaces, cost, placement)

```

```

VALUES (4, 0, 300, 5), (5, 0, 1000, 5), (6, 0, 3000, 5), (7, 0,
10000, 5), (8, 0, 30000, 5), (9, 0, 100000, 5), (10, 1, 300000, 5),
(11, 3, 1000000, 5), (12, 10, 3000000, 5), (13, 30, 10000000,
5), (14, 100, 30000000, 5), (15, 300, 100000000, 5)
GO
INSERT INTO habitats (sm, cabins, workspaces, cost, placement)
VALUES (6, 1, 0, 100, 5), (7, 2, 0, 300, 5), (8, 6, 0, 1000, 5),
(9, 20, 0, 3000, 5), (10, 60, 1, 10000, 5), (11, 200, 3, 30000,
5), (12, 600, 10, 100000, 5), (13, 2000, 30, 300000, 5), (14,
6000, 100, 1000000, 5), (15, 20000, 300, 3000000, 5)
GO
INSERT INTO life_support (name, cabin_modifier, min_tl)
VALUES ('Life support', 1, 7), ('Total life support', 2, 9)
INSERT INTO rooms (name, occupancy, cost, size, min_tl)
VALUES ('Bunkroom', 4, 0, 1, 0), ('Cabin', 2, 0, 1, 0), ('Cell', 4,
0, 1, 0), /*('Cage', 4, 0, 1, 0),*/ ('Luxury Cabin', 2, 0, 2, 0),
('Briefing Room', 0, 0, 1, 0), ('Establishment', 0, 0, 2, 0),
('Hibernation Chamber', 1, 0, 0.25, 9), ('Lab (Other)', 0, 1000,
2, 0), ('Lab (Physics)', 0, 10000, 2, 0), ('Lab (Science!)', 0,
30000, 2, 0), ('Minifac (Fabricator)', 0, 500, 1, 8),
('Minifac (Robofac)', 0, 1000, 1, 10), ('Minifac (Nanofactory)',
0, 2000, 1, 11), ('Minifac (Replicator)', 0, 10000, 1, 12),
('Office', 0, 0, 1, 0), ('Sickbay', 0, 0, 1, 0), ('Sickbay
(Automated)', 0, 100, 1, 0), ('Teleport Projector', 0, 20000, 1,
12), ('Teleport Projector (Send)', 0, 10000, 1, 12), ('Teleport
Projector (Receive)', 0, 10000, 1, 12)
GO
INSERT INTO fuel_tanks (sm, capacity, cost, placement, min_tl)
VALUES (4, 0.5, 3, 5, 7), (5, 1.5, 10, 5, 7), (6, 5, 30, 5, 7),
(7, 15, 100, 5, 7), (8, 50, 300, 5, 7), (9, 150, 1000, 5, 7), (10,
500, 3000, 5, 7), (11, 1500, 10000, 5, 7), (12, 5000, 30000, 5,
7), (13, 15000, 100000, 5, 7), (14, 50000, 300000, 5, 7),
(15, 150000, 1000000, 5, 7)
INSERT INTO fuel (name, cost_per_ton)
VALUES ('None', 0), ('Water', 20), ('Coolant', 150), ('Ionizable
reaction mass', 180), ('Rock dust', 2), ('Rocket Fuel', 800),
('Hydrogen', 2000),
('Jet Fuel', 4000), ('HEDM Rocket Fuel', 6000), ('Antimatter-
catalyzed hydrogen', 20000), ('Antimatter-catalyzed water',
20000),
('Nuclear pellets', 50000), ('Uranium-saltwater', 100000),
('Antimatter-boosted hydrogen', 12000000),
('Antimatter-boosted water', 12000000), ('Nuclear bomb pulse
units', 250000), ('Matter/antimatter', 1000000000000000)
GO

```

```

INSERT INTO engine_types (name, power_draw,
high_thrust_modifier, ram_rocket, placement)
VALUES ('Jet Engine', 0, 0, 0, 3), ('Chemical Rocket', 0, 0, 0,
3), ('HEDM Chemical Rocket', 0, 0, 0, 3), ('Ion Drive', 1, 0, 0,
3), ('Mass Driver', 1, 2, 0, 3), ('Nuclear Thermal Rocket', 0, 0,
1, 3), ('Nuclear Light Bulb', 0, 0, 0, 3), ('Nuclear Saltwater
Rocket', 0, 0, 0, 3), ('External Pulsed Plasma Drive', 0, 2, 0,
3), ('Fusion Pulse Drive', 0, 2, 0, 3), ('Advanced Fusion Pulse
Drive', 0, 2, 0, 3), ('Super Fusion Pulse Drive', 0, 2, 0, 3),
('Fusion Rocket', 0, 2, 0, 3), ('Fusion Torch', 0, 2, 1, 3),
('Super Fusion Torch', 0, 2, 1, 3), ('Antimatter Thermal
Rocket', 0, 5, 1, 3), ('Antimatter Plasma Rocket', 0, 5, 0, 3),
('Antimatter Plasma Torch', 0, 5, 1, 3), ('Super Antimatter Plasma
Torch', 0, 5, 1, 3), ('Antimatter Pion', 0, 5, 0, 3), ('Antimatter
Pion Torch', 0, 5, 0, 3), ('Total Conversion Torch', 0, 5, 0,
3), ('Super Conversion Torch', 0, 5, 0, 3), ('Rotary Reactionless
Drive', 1, 0, 0, 2), ('Rotary Reactionless Drive', 1, 0, 0, 3),
('Standard Reactionless Drive', 1, 0, 0, 3), ('Hot Reactionless
Drive', 1, 0, 0, 3), ('Super Reactionless Drive', 1, 0, 0, 3),
('Subwarp Drive', 1, 0, 0, 3), ('Lightsail', 0, 0, 0, 4),
('Magsail', 1, 0, 0, 4)
GO
INSERT INTO engine_variations (engine_type, delta_v_per_tank,
acceleration, fuel, min_t1)
VALUES (1, 0, 1, 7, 7), (2, 0.15, 3, 5, 7), (3, 0.5, 2, 8, 9),
(4, 3, 0.0005, 3, 8), (5, 0.3, 0.01, 4, 9), (6, 0.3, 0.1, 6, 7),
(6, 0.3, 0.2, 6, 8), (6, 0.3, 0.5, 6, 9), (6, 0.1, 0.3, 1, 7),
(6, 0.1, 0.6, 1, 8), (6, 0.1, 1.5, 1, 9), (7, 0.8, 0.01, 6, 9),
(7, 0.8, 0.05, 6, 10), (8, 2.5, 2, 12, 9), (9, 2, 2, 15, 7), (9,
3, 2, 15, 8), (9, 4, 2, 15, 9), (9, 8, 2, 15, 10), (10, 5, 0.01,
11, 9), (10, 10, 0.05, 11, 10), (10, 40, 0.05, 11, 11), (11, 20,
0.005, 11, 9), (11, 100, 0.005, 11, 10), (11, 350, 0.005, 11,
11), (12, 350, 20, 11, 11), (12, 350, 100, 11, 12), (13, 12,
0.005, 6, 9), (13, 60, 0.005, 6, 10), (13, 180, 0.005, 6, 11),
(13, 450, 0.005, 6, 12), (13, 4, 0.005, 1, 9), (13, 20, 0.005,
1, 10), (13, 60, 0.005, 1, 11), (13, 150, 0.005, 1, 12),
(14, 15, 0.5, 6, 10), (14, 45, 0.5, 6, 11), (14, 150, 0.5, 6,
12), (14, 5, 1.5, 1, 10), (14, 15, 1.5, 1, 11), (14, 50, 1.5, 1,
12), (15, 450, 50, 6, 11), (15, 150, 150, 1, 11), (16, 1.8, 0.1,
9, 9), (16, 1.8, 0.2, 9, 10), (16, 1.8, 0.4, 9, 11), (16, 0.6,
0.3, 10, 9), (16, 0.6, 0.6, 10, 10), (16, 0.6, 1.2, 10, 11), (17,
120, 0.01, 13, 10), (17, 360, 0.01, 13, 11), (17, 40, 0.03, 14,
10), (17, 120, 0.03, 14, 11), (18, 120, 1, 13, 10), (18, 360, 1,
13, 11), (18, 40, 3, 14, 10), (18, 120, 3, 14, 11), (19, 360,
100, 13, 11), (19, 120, 300, 14, 11), (20, 3400, 0.005, 16, 11),
(21, 3400, 0.1, 16, 11), (22, 100000, 1, 6, 12), (23, 10000, 50,

```

```

6, 12), (24, 0, 0.1, 0, 7), (25, 0, 0.1, 0, 7), (26, 0, 0.5, 0,
10), (26, 0, 1, 0, 11), (27, 0, 1, 0, 10), (27, 0, 2, 0, 11),
(28, 0, 50, 0, 11), (28, 0, 100, 0, 12), (29, 0, 500, 0, 9), (30,
0, 0.0001, 0, 9), (31, 0, 0.001, 0, 9)
GO
INSERT INTO engines (engine_type, sm, cost, workspaces)
VALUES (1, 4, 100, 0), (1, 5, 300, 0), (1, 6, 1000, 0), (1, 7,
3000, 0), (1, 8, 10000, 0), (1, 9, 30000, 0), (1, 10, 100000,
1), (1, 11, 300000, 3), (1, 12, 10000000, 10), (1, 13, 3000000,
30), (1, 14, 10000000, 100), (1, 15, 30000000, 300), (2, 4, 20,
0), (2, 5, 60, 0), (2, 6, 200, 0), (2, 7, 600, 0), (2, 8, 2000,
0), (2, 9, 6000, 0), (2, 10, 20000, 1), (2, 11, 60000, 3),
(2, 12, 200000, 10), (2, 13, 600000, 30), (2, 14, 2000000, 100),
(2, 15, 6000000, 600), (3, 4, 30, 0), (3, 5, 90, 0), (3, 6, 300,
0), (3, 7, 900, 0), (3, 8, 3000, 0), (3, 9, 9000, 0), (3, 10,
30000, 1), (3, 11, 90000, 3), (3, 12, 300000, 10), (3, 13,
900000, 30), (3, 14, 3000000, 100), (3, 15, 9000000, 600), (4, 4,
100, 0), (4, 5, 300, 0), (4, 6, 1000, 0), (4, 7, 3000, 0), (4,
8, 10000, 0), (4, 9, 30000, 0), (4, 10, 100000, 1), (4, 11,
300000, 3), (4, 12, 1000000, 10), (4, 13, 3000000, 30), (4, 14,
10000000, 100), (4, 15, 30000000, 300), (5, 4, 100, 0), (5, 5,
300, 0), (5, 6, 1000, 0), (5, 7, 3000, 0), (5, 8, 10000, 0), (5,
9, 30000, 0), (5, 10, 100000, 1), (5, 11, 300000, 3), (5, 12,
1000000, 10), (5, 13, 3000000, 30), (5, 14, 10000000, 100), (5,
15, 30000000, 300), (6, 4, 50, 0), (6, 5, 150, 0), (6, 6, 500,
0), (6, 7, 1500, 0), (6, 8, 5000, 0), (6, 9, 15000, 0), (6, 10,
50000, 0), (6, 11, 150000, 0), (6, 12, 500000, 0), (6, 13,
1500000, 0), (6, 14, 5000000, 0), (6, 15, 15000000, 0), (7, 4,
50, 0), (7, 5, 150, 0), (7, 6, 500, 0), (7, 7, 1500, 0), (7, 8,
5000, 0), (7, 9, 15000, 0), (7, 10, 50000, 0), (7, 11, 150000,
0), (7, 12, 500000, 0), (7, 13, 1500000, 0), (7, 14, 5000000, 0),
(7, 15, 15000000, 0), (8, 4, 50, 0), (8, 5, 150, 0), (8, 6, 500,
0), (8, 7, 1500, 0), (8, 8, 5000, 0), (8, 9, 15000, 0), (8, 10,
50000, 0), (8, 11, 150000, 0), (8, 12, 500000, 0), (8, 13,
1500000, 0), (8, 14, 5000000, 0), (8, 15, 15000000, 0), (9, 4,
100, 0), (9, 5, 300, 0), (9, 6, 1000, 0), (9, 7, 3000, 0), (9,
8, 10000, 0), (9, 9, 30000, 0),
(9, 10, 100000, 1), (9, 11, 300000, 3), (9, 12, 1000000, 10),
(9, 13, 3000000, 30), (9, 14, 10000000, 100), (9, 15, 30000000,
300), (10, 4, 100, 0), (10, 5, 300, 0), (10, 6, 1000, 0), (10, 7,
3000, 0), (10, 8, 10000, 0), (10, 9, 30000, 0), (10, 10, 100000,
1), (10, 11, 300000, 3), (10, 12, 1000000, 10), (10, 13,
3000000, 30), (10, 14, 10000000, 100), (10, 15, 30000000, 300),
(11, 4, 200, 0), (11, 5, 600, 0), (11, 6, 2000, 0), (11, 7, 6000,
0), (11, 8, 20000, 0), (11, 9, 60000, 0), (11, 10, 200000, 1),
(11, 11, 600000, 3), (11, 12, 2000000, 10), (11, 13, 6000000,

```

30), (11, 14, 20000000, 100), (11, 15, 60000000, 300), (12, 4, 400, 0), (12, 5, 1200, 0), (12, 6, 4000, 0), (11, 7, 12000, 0), (12, 8, 40000, 0), (12, 9, 120000, 0), (12, 10, 400000, 1), (12, 11, 1200000, 3), (12, 12, 40000000, 10), (12, 13, 12000000, 30), (12, 14, 40000000, 100), (12, 15, 120000000, 300), (13, 4, 100, 0), (13, 5, 300, 0), (13, 6, 1000, 0), (13, 7, 3000, 0), (13, 8, 10000, 0), (13, 9, 30000, 0), (13, 10, 100000, 1), (13, 11, 300000, 3), (13, 12, 1000000, 3), (13, 13, 3000000, 3), (13, 14, 10000000, 3), (13, 15, 30000000, 3), (14, 4, 200, 0), (14, 5, 600, 0), (14, 6, 2000, 0), (14, 7, 6000, 0), (14, 8, 20000, 0), (14, 9, 60000, 0), (14, 10, 200000, 1), (14, 11, 600000, 3), (14, 12, 2000000, 3), (14, 13, 6000000, 3), (14, 14, 20000000, 3), (14, 15, 60000000, 3), (15, 4, 400, 0), (15, 5, 1200, 0), (15, 6, 4000, 0), (15, 7, 12000, 0), (15, 8, 40000, 0), (15, 9, 120000, 0), (15, 10, 400000, 1), (15, 11, 1200000, 3), (15, 12, 4000000, 3), (15, 13, 12000000, 3), (15, 14, 40000000, 3), (15, 15, 120000000, 3), (16, 4, 50, 0), (16, 5, 150, 0), (16, 6, 500, 0), (16, 7, 1500, 0), (16, 8, 5000, 0), (16, 9, 15000, 0), (16, 10, 50000, 1), (16, 11, 150000, 3), (16, 12, 500000, 10), (16, 13, 1500000, 30), (16, 14, 5000000, 100), (16, 15, 15000000, 300), (17, 4, 50, 0), (17, 5, 150, 0), (17, 6, 500, 0), (17, 7, 1500, 0), (17, 8, 5000, 0), (17, 9, 15000, 0), (17, 10, 50000, 1), (17, 11, 150000, 3), (17, 12, 500000, 10), (17, 13, 1500000, 30), (17, 14, 5000000, 100), (17, 15, 15000000, 300), (18, 4, 100, 0), (18, 5, 300, 0), (18, 6, 1000, 0), (18, 7, 3000, 0), (18, 8, 10000, 0), (18, 9, 30000, 0), (18, 10, 100000, 1), (18, 11, 300000, 3), (18, 12, 1000000, 10), (18, 13, 3000000, 30), (18, 14, 10000000, 100), (18, 15, 30000000, 300), (19, 4, 200, 0), (19, 5, 600, 0), (19, 6, 2000, 0), (19, 7, 6000, 0), (19, 8, 20000, 0), (19, 9, 60000, 0), (19, 10, 200000, 1), (19, 11, 600000, 3), (19, 12, 2000000, 10), (19, 13, 6000000, 30), (19, 14, 20000000, 100), (19, 15, 60000000, 300), (20, 4, 100, 0), (20, 5, 300, 0), (20, 6, 1000, 0), (20, 7, 3000, 0), (20, 8, 10000, 0), (20, 9, 30000, 0), (20, 10, 100000, 1), (20, 11, 300000, 3), (20, 12, 1000000, 10), (20, 13, 3000000, 30), (20, 14, 10000000, 100), (20, 15, 30000000, 300), (21, 4, 200, 0), (21, 5, 600, 0), (21, 6, 2000, 0), (21, 7, 6000, 0), (21, 8, 20000, 0), (21, 9, 60000, 0), (21, 10, 200000, 1), (21, 11, 600000, 3), (21, 12, 2000000, 10), (21, 13, 6000000, 30), (21, 14, 20000000, 100), (21, 15, 60000000, 300), (22, 4, 200, 0), (22, 5, 600, 0), (22, 6, 2000, 0), (22, 7, 6000, 0), (22, 8, 20000, 0), (22, 9, 60000, 0), (22, 10, 200000, 1), (22, 11, 600000, 3), (22, 12, 2000000, 10), (22, 13, 6000000, 30), (22, 14, 20000000, 100), (22, 15, 60000000, 300), (23, 4, 500, 0), (23, 5, 1500, 0), (23, 6, 5000, 0), (23, 7, 15000, 0), (23, 8, 50000, 0), (23, 9, 150000, 0), (23, 10, 500000, 1), (23, 11,

```

1500000, 3), (23, 12, 5000000, 10), (23, 13, 15000000, 30), (23,
14, 50000000, 100), (23, 15, 150000000, 300), (24, 4, 5, 0),
(24, 5, 15, 0), (24, 6, 50, 0), (24, 7, 150, 0), (24, 8, 500, 0),
(24, 9, 1500, 0), (24, 10, 5000, 1), (24, 11, 15000, 3), (24,
12, 50000, 10), (24, 13, 150000, 30), (24, 14, 500000, 100), (24,
15, 1500000, 300), (25, 4, 10, 0), (25, 5, 30, 0), (25, 6, 100,
0), (25, 7, 300, 0), (25, 8, 1000, 0), (25, 9, 3000, 0), (25, 10,
10000, 1), (25, 11, 30000, 3), (25, 12, 100000, 10), (25, 13,
300000, 30), (25, 14, 1000000, 100), (25, 15, 3000000, 300), (26,
4, 10, 0), (26, 5, 30, 0), (26, 6, 100, 0), (26, 7, 300, 0),
(26, 8, 1000, 0), (26, 9, 3000, 0), (26, 10, 10000, 1), (26, 11,
30000, 3), (26, 12, 100000, 10), (26, 13, 300000, 30), (26, 14,
1000000, 100), (26, 15, 3000000, 300), (27, 4, 60, 0), (27, 5,
200, 0), (27, 6, 600, 0), (27, 7, 2000, 0), (27, 8, 6000, 0),
(27, 9, 20000, 0), (27, 10, 60000, 1), (27, 11, 200000, 3), (27,
12, 600000, 10), (27, 13, 2000000, 30), (27, 14, 6000000, 100),
(27, 15, 20000000, 300), (28, 4, 100, 0), (28, 5, 300, 0), (28,
6, 1000, 0), (28, 7, 3000, 0), (28, 8, 10000, 0), (28, 9, 30000,
0), (28, 10, 100000, 1), (28, 11, 300000, 3), (28, 12, 1000000,
10), (28, 13, 3000000, 30), (28, 14, 10000000, 100), (28, 15,
30000000, 300), (29, 4, 100, 0), (29, 5, 300, 0), (29, 6, 1000,
0), (29, 7, 3000, 0), (29, 8, 10000, 0), (29, 9, 30000, 0), (29,
10, 100000, 1), (29, 11, 300000, 3), (29, 12, 1000000, 3), (30,
4, 100, 0), (30, 5, 300, 0), (30, 6, 1000, 0), (30, 7, 3000, 0),
(30, 8, 10000, 0), (30, 9, 30000, 0), (30, 10, 100000, 1), (30,
11, 300000, 3), (30, 12, 1000000, 3)

```

GO

```

INSERT INTO armor_types (name, min_tl)
VALUES ('Ice', 0), ('Stone', 0), ('Steel', 7), ('Light Alloy',
7), ('Metallic Laminate', 8), ('Advanced Metallic Laminate', 9),
('Nanocomposite', 10), ('Organic', 10), ('Diamondoid', 11),
('Exotic Laminate', 12)

```

GO

```

INSERT INTO armor_systems (sm, armor_type, streamlined_dr,
unstreamlined_dr, cost, placement)
VALUES (8, 1, 0, 1, 0, 4), (9, 1, 0, 2, 0, 4), (10, 1, 0, 2, 0,
4), (11, 1, 0, 3, 0, 4), (12, 1, 0, 5, 0, 4), (13, 1, 0, 7, 0,
4), (14, 1, 0, 10, 0, 4), (15, 1, 0, 15, 0, 4), (7, 2, 0, 1, 0,
4), (8, 2, 0, 2, 0, 4), (9, 2, 0, 2, 0, 4), (10, 2, 0, 3, 0, 4),
(11, 2, 0, 5, 0, 4), (12, 2, 0, 7, 0, 4), (13, 2, 0, 10, 0, 4),
(14, 2, 0, 15, 0, 4), (15, 2, 0, 20, 0, 4), (5, 3, 0, 1, 6, 4),
(6, 3, 1, 2, 20, 4), (7, 3, 2, 3, 60, 4), (8, 3, 3, 5, 200, 4),
(9, 3, 5, 7, 600, 4), (10, 3, 7, 10, 2000, 4), (11, 3, 10, 15,
6000, 4), (12, 3, 15, 20, 20000, 4), (13, 3, 20, 30, 60000, 4),
(14, 3, 30, 50, 200000, 4), (15, 3, 50, 70, 600000, 4), (4, 4,
1, 1, 5, 4), (5, 4, 1, 2, 15, 4), (6, 4, 2, 3, 50, 4), (7, 4, 3,

```

```

5, 150, 4), (8, 4, 5, 7, 500, 4), (9, 4, 7, 10, 1500, 4), (10,
4, 10, 15, 5000, 4), (11, 4, 15, 20, 15000, 4), (12, 4, 20, 30,
50000, 4), (13, 4, 30, 50, 150000, 4), (14, 4, 50, 70, 500000,
4), (15, 4, 70, 100, 1500000, 4), (4, 5, 1, 2, 10, 4), (5, 5, 2,
3, 30, 4), (6, 5, 3, 5, 100, 4), (7, 5, 5, 7, 300, 4), (8, 5, 7,
10, 1000, 4), (9, 5, 10, 15, 3000, 4), (10, 5, 15, 20, 10000, 4),
(11, 5, 20, 30, 30000, 4), (12, 5, 30, 50, 100000, 4), (13, 5,
50, 70, 300000, 4), (14, 5, 70, 100, 1000000, 4), (15, 5, 100,
150, 3000000, 4), (4, 6, 2, 3, 20, 4), (5, 6, 3, 5, 60, 4), (6,
6, 5, 7, 200, 4), (7, 6, 7, 10, 600, 4),
(8, 6, 10, 15, 2000, 4), (9, 6, 15, 20, 6000, 4), (10, 6, 20,
30, 20000, 4), (11, 6, 30, 50, 60000, 4), (12, 6, 50, 70,
200000, 4), (13, 6, 70, 100, 600000, 4), (14, 6, 100, 150,
2000000, 4), (15, 6, 150, 200, 6000000, 4), (4, 7, 3, 5, 50, 4),
(5, 7, 5, 7, 150, 4), (6, 7, 7, 10, 500, 4), (7, 7, 10, 15, 1500,
4), (8, 7, 15, 20, 5000, 4), (9, 7, 20, 30, 15000, 4), (10, 7,
30, 50, 50000, 4),
(11, 7, 50, 70, 150000, 4), (12, 7, 70, 100, 500000, 4), (13, 7,
100, 150, 1500000, 4), (14, 7, 150, 200, 5000000, 4),
(15, 7, 200, 300, 15000000, 4), (4, 8, 0, 1, 3, 4), (5, 8, 1, 2,
10, 4), (6, 8, 2, 3, 30, 4), (7, 8, 3, 5, 100, 4), (8, 8, 5, 7,
300, 4), (9, 8, 7, 10, 1000, 4), (10, 8, 10, 15, 3000, 4), (11,
8, 15, 20, 10000, 4), (12, 8, 20, 30, 30000, 4), (13, 8, 30, 50,
100000, 4), (14, 8, 50, 70, 300000, 4), (15, 8, 70, 100, 1000000,
4), (4, 9, 5, 7, 100, 4), (5, 9, 7, 10, 300, 4), (6, 9, 10, 15,
1000, 4), (7, 9, 15, 20, 3000, 4), (8, 9, 20, 30, 10000, 4), (9,
9, 30, 50, 30000, 4), (10, 9, 50, 70, 100000, 4), (11, 9, 70,
100, 300000, 4), (12, 9, 100, 150, 1000000, 4), (13, 9, 150, 200,
3000000, 4), (14, 9, 200, 300, 10000000, 4), (15, 9, 300, 500,
30000000, 4), (4, 10, 7, 10, 200, 4), (5, 10, 10, 15, 600, 4),
(6, 10, 15, 20, 2000, 4), (7, 10, 20, 30, 6000, 4), (8, 10, 30,
50, 20000, 4), (9, 10, 70, 50, 60000, 4), (10, 10, 70, 100,
200000, 4), (11, 10, 100, 150, 600000, 4), (12, 10, 150, 200,
2000000, 4), (13, 10, 200, 300, 6000000, 4), (14, 10, 300, 500,
20000000, 4), (15, 10, 500, 700, 60000000, 4)

```

GO

```

INSERT INTO hangar_bays (sm, capacity, launch_rate, workspaces,
cost, min_tl, placement)
VALUES (5, 1, 1, 0, 3, 7, 4), (6, 3, 3, 0, 10, 7, 4), (7, 10,
10, 0, 30, 7, 4), (8, 30, 20, 0, 100, 7, 4), (9, 100, 50, 0,
300, 7, 4), (10, 300, 100, 1, 1000, 7, 4), (11, 1000, 200, 3,
3000, 7, 4), (12, 3000, 500, 10, 10000, 7, 4), (13, 10000, 1000,
30, 30000, 7, 4), (14, 30000, 2000, 100, 100000, 7, 4), (15,
100000, 5000, 100, 300000, 7, 4)

```

GO

```

INSERT INTO sensor_arrays (sm, array_level, workspaces, cost,
min_tl, placement)
VALUES (4, -5, 0, 20, 7, 4), (5, -4, 0, 60, 7, 4), (6, -3, 0,
200, 7, 4), (7, -2, 0, 600, 7, 4), (8, -1, 0, 2000, 7, 4),
(9, 0, 0, 6000, 7, 4), (10, 1, 1, 20000, 7, 4), (11, 2, 3,
60000, 7, 4), (12, 3, 10, 200000, 7, 4), (13, 4, 30, 600000, 7,
4), (14, 5, 100, 2000000, 7, 4), (15, 6, 300, 6000000, 7, 4)
GO
INSERT INTO sensor_array_types (name, cost_multiplier)
VALUES ('Enhanced', 1), ('Science', 5), ('Tactical', 5),
('Multipurpose', 10)
GO
INSERT INTO cargo_holds (sm, capacity, placement)
VALUES (4, 0.5, 5), (5, 1.5, 5), (6, 5, 5), (7, 15, 5), (8, 50,
5), (9, 150, 5), (10, 500, 5), (11, 1500, 5), (12, 5000, 5),
(13, 15000, 5), (14, 50000, 5), (15, 150000, 5)
GO
INSERT INTO ramscoops (sm, workspaces, cost, min_tl, placement)
VALUES (4, 0, 1000, 10, 1), (5, 0, 3000, 10, 1), (6, 0, 10000,
10, 1), (7, 0, 30000, 10, 1), (8, 0, 100000, 10, 1), (9, 0,
300000, 10, 1), (10, 1, 1000000, 10, 1), (11, 3, 3000000, 10, 1),
(12, 10, 10000000, 10, 1), (13, 30, 30000000, 10, 1), (14, 100,
100000000, 10, 1), (15, 300, 300000000, 10, 1)
GO
INSERT INTO power_plant_types (name, power_points, deratable,
endurance_unit, placement)
VALUES ('Fuel Cell', 1, 0, 'h', 5), ('MHD Turbine', 2, 0, 'h',
5), ('Fission Reactor', 1, 0, 'y', 5), ('Fusion Reactor', 2, 1,
'y', 5), ('Antimatter Reactor', 4, 1, 'y', 5), ('Super Fusion
Reactor', 4, 1, 'y', 5), ('Total Conversion Reactor', 5, 0, 'y',
5), ('Solar Panel Array', 1, 0, '', 4)
GO
INSERT INTO power_plant_variants (power_plant_type, endurance,
min_tl)
VALUES (1, 3, 7), (1, 6, 8), (1, 12, 9), (1, 24, 10), (2, 6, 9),
(2, 12, 10), (3, 25, 8), (3, 50, 9), (3, 75, 10), (4, 50, 9),
(4, 200, 10), (4, 600, 11), (4, 1500, 12), (5, 2, 10), (5, 20,
11), (5, 200, 12), (6, 400, 11), (6, 1000, 12), (7, 0, 12), (8,
0, 7)
GO
INSERT INTO power_plants (power_plant_type, sm, workspaces,
cost)
VALUES (1, 4, 0, 5), (1, 5, 0, 15), (1, 6, 0, 50), (1, 7, 0,
150), (1, 8, 0, 500), (1, 9, 0, 1500), (1, 10, 1, 5000), (1, 11,
3, 15000), (1, 12, 10, 50000), (1, 13, 30, 150000), (1, 14, 100,
500000), (1, 15, 300, 1500000), (2, 4, 0, 10), (2, 5, 0, 30),

```

```
(2, 6, 0, 100), (2, 7, 0, 300), (2, 8, 0, 1000), (2, 9, 0, 3000),
(2, 10, 1, 10000), (2, 11, 3, 30000), (2, 12, 10, 100000), (2,
13, 30, 300000), (2, 14, 100, 1000000), (2, 15, 300, 3000000),
(3, 4, 0, 30), (3, 5, 0, 100), (3, 6, 0, 300), (3, 7, 0, 1000),
(3, 8, 0, 3000), (3, 9, 0, 10000), (3, 10, 1, 30000), (3, 11, 3,
100000), (3, 12, 10, 300000), (3, 13, 30, 1000000), (3, 14, 100,
3000000), (3, 15, 300, 10000000), (4, 4, 0, 100), (4, 5, 0, 300),
(4, 6, 0, 1000), (4, 7, 0, 3000), (4, 8, 0, 10000), (4, 9, 0,
30000), (4, 10, 1, 100000), (4, 11, 3, 300000), (4, 12, 10,
1000000), (4, 13, 30, 3000000), (4, 14, 100, 10000000), (4, 15,
300, 30000000), (5, 4, 0, 200), (5, 5, 0, 600), (5, 6, 0, 2000),
(5, 7, 0, 6000), (5, 8, 0, 20000), (5, 9, 0, 60000), (5, 10, 1,
200000), (5, 11, 3, 600000), (5, 12, 10, 2000000), (5, 13, 30,
6000000), (5, 14, 100, 20000000), (5, 15, 300, 60000000), (6, 4,
0, 300), (6, 5, 0, 1000), (6, 6, 0, 3000), (6, 7, 0, 10000), (6,
8, 0, 30000), (6, 9, 0, 100000), (6, 10, 1, 300000), (6, 11, 3,
1000000), (6, 12, 10, 3000000), (6, 13, 30, 10000000), (6, 14,
100, 30000000), (6, 15, 300, 100000000), (7, 4, 0, 600), (7, 5,
0, 2000), (7, 6, 0, 6000), (7, 7, 0, 20000), (7, 8, 0, 60000),
(7, 9, 0, 200000), (7, 10, 1, 600000), (7, 11, 3, 2000000), (7,
12, 10, 6000000), (7, 13, 30, 20000000), (7, 14, 100, 60000000),
(7, 15, 300, 200000000), (8, 4, 0, 50), (8, 5, 0, 150), (8, 6,
0, 500), (8, 7, 0, 1500), (8, 8, 0, 5000), (8, 9, 0, 15000), (8,
10, 0, 50000), (8, 11, 0, 150000), (8, 12, 0, 500000), (8, 13,
0, 1500000), (8, 14, 0, 5000000), (8, 15, 0, 15000000)
```

GO

```
INSERT INTO passenger_seating (sm, seats, cost, placement)
VALUES (4, 1, 3, 5), (5, 2, 10, 5), (6, 6, 30, 5), (7, 20, 100,
5), (8, 60, 300, 5), (9, 200, 1000, 5), (10, 600, 3000, 5),
(11, 2000, 10000, 5), (12, 6000, 30000, 5), (13, 20000, 100000,
5), (14, 60000, 300000, 5), (15, 200000, 1000000, 5)
```

GO

```
INSERT INTO weapon_types(name)
VALUES ('Beam'), ('Gun'), ('Missile')
```

GO

```
INSERT INTO weapon_subtypes(name, weapon_type, power_draw,
damage_level, armor_divisor, damage_types, min_tl)
VALUES('Heat ray', 1, 1, 1, 1, 'burn', 7), ('Laser', 1, 1, 1, 2,
'burn', 9), ('Particle', 1, 1, 1, 5, 'burn rad sur', 10),
('Plasma', 1, 1, 2, 2, 'burn exp', 10), ('UV Laser', 1, 1, 1, 2,
'burn', 10), ('Antiparticle', 1, 1, 2, 3, 'cr exp sur rad', 11),
('Ghost Particle', 1, 1, 1, 0, 'cr exp', 11), ('Graviton', 1, 1,
1, 0, 'cr', 11), ('Tractor', 1, 1, 1, 1, 'special', 11),
('X-ray Laser', 1, 1, 1, 5, 'burn sur', 11), ('Conversion', 1,
1, 1, 10, 'cor, fw. burn exp rad sur', 12),
```

```

('Disintegrator', 1, 1, 1, 0, 'cor', 12), ('Graser', 1, 1, 1,
10, 'burn sur', 12), ('Conventional Gun', 2, 0, 0, 2, 'cr', 7),
('Electromagnetic Gun', 2, 1, 0, 2, 'cr', 9), ('Grav Gun', 2, 1,
0, 2, 'cr', 9), ('Missile Launcher', 3, 0, 0, 2, 'cr', 7),
('Warp Missile Launchers', 3, 1, 0, 2, 'cr', 11)
GO
INSERT INTO weapon_placements(name)
VALUES ('Fixed Mount'), ('Turret'), ('Spinal mount')
GO
INSERT INTO rates_of_fire(name, rate_of_fire)
VALUES ('Standard', 1), ('Rapid Fire', 10), ('Very Rapid Fire',
100)
GO
INSERT INTO weapon_system_types(name, placement)
VALUES ('Major', 4), ('Medium', 4), ('Secondary', 4),
('Tertiary', 4), ('Spinal', 6)
GO
INSERT INTO weapon_calibers(caliber, damage, weight_per_shot)
VALUES (2, '6x5', 0.00025), (2.5, '6x6', 0.0005), (3, '6x7',
0.000625), (3.5, '6x8', 0.0008333), (4, '6x10', 0.00125), (5,
'6x12', 0.0025),
(6, '6x15', 0.005), (7, '6x18', 0.008), (8, '6x20', 0.0125),
(10, '6x25', 0.025), (12, '6x30', 0.04), (14, '6x35', 0.0625),
(16, '6x40', 0.1), (20, '6x50', 0.2), (24, '6x60', 0.3333), (28,
'6x70', 0.5), (32, '6x80', 1), (40, '6x100', 2), (48, '6x120',
4), (56, '6x140', 6), (64, '6x160', 7.5), (80, '6x200', 15), (96,
'6x240', 25), (112, '6x280', 40)
GO
INSERT INTO weapon_power(power_output, damage1, damage2)
VALUES (3, '1d-4', '1d-2'), (10, '1d-3', '1d-1'), (30, '1d-2',
'1d+1'), (100, '1d', '2d'), (300, '1d+2', '3d'), (1000, '2d',
'4d'),
(3000, '3d', '6d'), (10000, '4d', '8d'), (30000, '6d', '6dx2'),
(100000, '2dx5', '4dx5'), (300000, '3dx5', '3dx10'),
(1000000, '4dx5', '4dx10'), (3000000, '3dx10', '6dx10'),
(10000000, '4dx10', '8dx10'), (30000000, '6dx10', '6dx20'),
(100000000, '2dx50', '2dx100'), (300000000, '3dx50', '3dx100'),
(1000000000, '2dx100', '4dx100'), (3000000000, '3dx100',
'6dx100'),
(10000000000, '4dx100', '8dx100'), (30000000000, '6dx100',
'6dx200'), (100000000000, '2dx500', '2dx1,000'),
(300000000000, '3dx500', '3dx1,000'), (1000000000000,
'2dx1,000', '4dx1,000'), (3000000000000, '3dx1,000', '6dx1,000')
GO
INSERT INTO beam_ranges(subtype, beam_power, half_range,
max_range)

```

VALUES (1, 1, 150, 500), (1, 2, 150, 500), (1, 3, 150, 500), (1, 4, 300, 1000), (1, 5, 300, 1000), (1, 6, 300, 1000), (1, 7, 700, 2000), (1, 8, 700, 2000), (1, 9, 700, 2000), (1, 10, 1500, 5000), (1, 11, 1500, 5000), (1, 12, 1500, 5000), (1, 13, 3000, 10000), (1, 14, 3000, 10000), (1, 15, 3000, 10000), (1, 16, 7000, 20000), (1, 17, 7000, 20000), (1, 18, 7000, 20000), (1, 19, 15000, 50000), (1, 20, 1500, 50000), (1, 21, 1500, 50000), (1, 22, 30000, 100000), (1, 23, 30000, 100000), (1, 24, 30000, 100000), (1, 25, 50000, 150000), (2, 1, 150, 500), (2, 2, 150, 500), (2, 3, 150, 500), (2, 4, 300, 1000), (2, 5, 300, 1000), (2, 6, 300, 1000), (2, 7, 700, 2000), (2, 8, 700, 2000), (2, 9, 700, 2000), (2, 10, 1500, 5000), (1, 11, 1500, 5000), (2, 12, 1500, 5000), (2, 13, 3000, 10000), (2, 14, 3000, 10000), (2, 15, 3000, 10000), (2, 16, 7000, 20000), (1, 17, 7000, 20000), (2, 18, 7000, 20000), (2, 19, 15000, 50000), (2, 20, 1500, 50000), (2, 21, 1500, 50000), (2, 22, 30000, 100000), (2, 23, 30000, 100000), (2, 24, 30000, 100000), (2, 25, 50000, 150000), (3, 1, 70, 200), (3, 2, 70, 200), (3, 3, 70, 200), (3, 4, 150, 500), (3, 5, 150, 500), (3, 6, 150, 500), (3, 7, 300, 1000), (3, 8, 300, 1000), (3, 9, 300, 1000), (3, 10, 700, 2000), (3, 11, 700, 2000), (3, 12, 700, 2000), (3, 13, 1500, 5000), (3, 14, 1500, 5000), (3, 15, 1500, 5000), (3, 16, 3000, 10000), (3, 17, 3000, 10000), (3, 18, 3000, 10000), (3, 19, 7000, 20000), (3, 20, 7000, 20000), (3, 21, 7000, 20000), (3, 22, 15000, 50000), (3, 23, 15000, 50000), (3, 24, 15000, 50000), (3, 25, 30000, 70000), (4, 1, 30, 100), (4, 2, 30, 100), (4, 3, 30, 100), (4, 4, 70, 200), (4, 5, 70, 200), (4, 6, 70, 200), (4, 7, 150, 500), (4, 8, 150, 500), (4, 9, 150, 500), (4, 10, 300, 1000), (4, 11, 300, 1000), (4, 12, 300, 1000), (4, 13, 700, 2000), (4, 14, 700, 2000), (4, 15, 700, 2000), (4, 16, 1500, 5000), (4, 17, 1500, 5000), (4, 18, 1500, 5000), (4, 19, 3000, 10000), (4, 20, 3000, 10000), (4, 21, 3000, 10000), (4, 22, 7000, 20000), (4, 23, 7000, 20000), (4, 24, 7000, 20000), (4, 25, 15000, 50000), (5, 1, 300, 1000), (5, 2, 300, 1000), (5, 3, 300, 1000), (5, 4, 700, 2000), (5, 5, 700, 2000), (5, 6, 700, 2000), (5, 7, 1500, 5000), (5, 8, 1500, 5000), (5, 9, 1500, 5000), (5, 10, 3000, 10000), (5, 11, 3000, 10000), (5, 12, 3000, 10000), (5, 13, 7000, 20000), (5, 14, 7000, 20000), (5, 15, 7000, 20000), (5, 16, 15000, 50000), (5, 17, 15000, 50000), (5, 18, 15000, 50000), (5, 19, 30000, 100000), (5, 20, 30000, 100000), (5, 21, 30000, 100000), (5, 22, 70000, 200000), (5, 23, 70000, 200000), (5, 24, 70000, 200000), (5, 25, 150000, 500000), (6, 1, 70, 200), (6, 2, 70, 200), (6, 3, 70, 200), (6, 4, 150, 500), (6, 5, 150, 500), (6, 6, 150, 500), (6, 7, 300, 1000),

(6, 8, 300, 1000), (6, 9, 300, 1000), (6, 10, 700, 2000), (6, 11, 700, 2000), (6, 12, 700, 2000), (3, 13, 1500, 5000), (6, 14, 1500, 5000), (6, 15, 1500, 5000), (6, 16, 3000, 10000), (6, 17, 3000, 10000), (6, 18, 3000, 10000), (6, 19, 7000, 20000), (6, 20, 7000, 20000), (6, 21, 7000, 20000), (6, 22, 15000, 50000), (6, 23, 15000, 50000), (6, 24, 15000, 50000), (6, 25, 30000, 70000), (7, 1, 70, 200), (7, 2, 70, 200), (7, 3, 70, 200), (7, 4, 150, 500), (7, 5, 150, 500), (7, 6, 150, 500), (7, 7, 300, 1000), (7, 8, 300, 1000), (7, 9, 300, 1000), (7, 10, 700, 2000), (7, 11, 700, 2000), (7, 12, 700, 2000), (7, 13, 1500, 5000), (7, 14, 1500, 5000), (7, 15, 1500, 5000), (7, 16, 3000, 10000), (7, 17, 3000, 10000), (7, 18, 3000, 10000), (7, 19, 7000, 20000), (7, 20, 7000, 20000), (7, 21, 7000, 20000), (7, 22, 15000, 50000), (7, 23, 15000, 50000), (7, 24, 15000, 50000), (7, 25, 30000, 70000), (8, 1, 70, 200), (8, 2, 70, 200), (8, 3, 70, 200), (8, 4, 150, 500), (8, 5, 150, 500), (8, 6, 150, 500), (8, 7, 300, 1000), (8, 8, 300, 1000), (8, 9, 300, 1000), (8, 10, 700, 2000), (8, 11, 700, 2000), (8, 12, 700, 2000), (8, 13, 1500, 5000), (8, 14, 1500, 5000), (8, 15, 1500, 5000), (8, 16, 3000, 10000), (8, 17, 3000, 10000), (8, 18, 3000, 10000), (8, 19, 7000, 20000), (8, 20, 7000, 20000), (8, 21, 7000, 20000), (8, 22, 15000, 50000), (8, 23, 15000, 50000), (8, 24, 15000, 50000), (8, 25, 30000, 70000), (9, 1, 70, 200), (9, 2, 70, 200), (9, 3, 70, 200), (9, 4, 150, 500), (9, 5, 150, 500), (9, 6, 150, 500), (9, 7, 300, 1000), (9, 8, 300, 1000), (9, 9, 300, 1000), (9, 10, 700, 2000), (9, 11, 700, 2000), (9, 12, 700, 2000), (9, 13, 1500, 5000), (9, 14, 1500, 5000), (9, 15, 1500, 5000), (9, 16, 3000, 10000), (9, 17, 3000, 10000), (9, 18, 3000, 10000), (9, 19, 7000, 20000), (9, 20, 7000, 20000), (9, 21, 7000, 20000), (9, 22, 15000, 50000), (9, 23, 15000, 50000), (9, 24, 15000, 50000), (9, 25, 30000, 70000), (10, 1, 300, 1000), (10, 2, 300, 1000), (10, 3, 300, 1000), (10, 4, 700, 2000), (10, 5, 700, 2000), (10, 6, 700, 2000), (10, 7, 1500, 5000), (10, 8, 1500, 5000), (10, 9, 1500, 5000), (10, 10, 3000, 10000), (10, 11, 3000, 10000), (10, 12, 3000, 10000), (10, 13, 7000, 20000), (10, 14, 7000, 20000), (10, 15, 7000, 20000), (10, 16, 15000, 50000), (10, 17, 15000, 50000), (10, 18, 15000, 50000), (10, 19, 30000, 100000), (10, 20, 30000, 100000), (10, 21, 30000, 100000), (10, 22, 70000, 200000), (10, 23, 70000, 200000), (10, 24, 70000, 200000), (10, 25, 150000, 500000), (11, 1, 150, 500), (11, 2, 150, 500), (11, 3, 150, 500), (11, 4, 300, 1000), (11, 5, 300, 1000), (11, 6, 300, 1000), (11, 7, 700, 2000), (11, 8, 700, 2000), (11, 9, 700, 2000), (11, 10, 1500, 5000), (11, 11, 1500, 5000), (11, 12, 1500, 5000), (11, 13, 3000, 10000), (11, 14, 3000, 10000), (11, 15, 3000, 10000), (11, 16, 7000, 20000), (11, 17, 7000, 20000), (11, 18, 7000, 20000), (11, 19, 15000, 50000),

```
(11, 20, 1500, 50000), (11, 21, 1500, 50000), (11, 22, 30000,
100000), (11, 23, 30000, 100000), (11, 24, 30000, 100000),
(11, 25, 50000, 150000), (12, 1, 150, 500), (12, 2, 150, 500),
(12, 3, 150, 500), (12, 4, 300, 1000), (12, 5, 300, 1000), (12,
6, 300, 1000), (12, 7, 700, 2000), (12, 8, 700, 2000), (12, 9,
700, 2000), (12, 10, 1500, 5000), (12, 11, 1500, 5000), (12, 12,
1500, 5000), (12, 13, 3000, 10000), (12, 14, 3000, 10000), (12,
15, 3000, 10000), (12, 16, 7000, 20000), (12, 17, 7000, 20000),
(12, 18, 7000, 20000), (12, 19, 15000, 50000),
(12, 20, 1500, 50000), (12, 21, 1500, 50000), (12, 22, 30000,
100000), (12, 23, 30000, 100000), (12, 24, 30000, 100000),
(12, 25, 50000, 150000), (13, 1, 300, 1000), (13, 2, 300, 1000),
(13, 3, 300, 1000), (13, 4, 700, 2000), (13, 5, 700, 2000), (13,
6, 700, 2000), (13, 7, 1500, 5000), (13, 8, 1500, 5000), (13, 9,
1500, 5000), (13, 10, 3000, 10000), (13, 11, 3000, 10000), (13,
12, 3000, 10000), (13, 13, 7000, 20000), (13, 14, 7000, 20000),
(13, 15, 7000, 20000), (13, 16, 15000, 50000), (13, 17, 15000,
50000), (13, 18, 15000, 50000), (13, 19, 30000, 100000), (13, 20,
30000, 100000), (13, 21, 30000, 100000), (13, 22, 70000,
200000), (13, 23, 70000, 200000), (13, 24, 70000, 200000),
(13, 25, 150000, 500000)
```

GO

```
INSERT INTO weapon_system_configurations(weapon_system_type,
weapon_count, sm, default_caliber, default_power,
shots_per_weapon,
weight_per_weapon, cost, workspaces)
```

```
VALUES (1, 1, 4, 9, 7, 50, 0, 100, 0), (1, 1, 5, 10, 8, 70, 0,
150, 0), (1, 1, 6, 11, 9, 100, 0, 600, 0), (1, 1, 7, 12, 10, 150,
0, 1500, 0), (1, 1, 8, 13, 11, 200, 0, 6000, 0), (1, 1, 9, 14,
12, 300, 0, 15000, 0), (1, 1, 10, 15, 13, 500, 0, 60000, 1), (1,
1, 11, 16, 14, 700, 0, 150000, 3), (1, 1, 12, 17, 15, 1000, 0,
600000, 10), (1, 1, 13, 18, 16, 1500, 0, 1500000, 30), (1, 1, 14,
19, 17, 2000, 0, 6000000, 100), (1, 1, 15, 20, 18, 3000, 0,
15000000, 300), (2, 3, 5, 9, 7, 50, 0.5, 150, 0), (2, 3, 6, 11,
8, 70, 1.5, 600, 0), (2, 3, 7, 12, 9, 100, 5, 1500, 0),
(2, 3, 8, 13, 10, 150, 15, 6000, 0), (2, 3, 9, 14, 11, 200, 50,
15000, 0), (2, 3, 10, 15, 12, 300, 150, 60000, 1),
(2, 3, 11, 16, 13, 500, 500, 150000, 3), (2, 3, 12, 17, 14, 700,
1500, 600000, 10), (2, 3, 13, 18, 15, 1000, 5000, 1500000, 30),
(2, 3, 14, 19, 16, 2000, 15000, 6000000, 100), (2, 3, 15, 20,
17, 3000, 50000, 15000000, 300), (3, 10, 6, 9, 7, 50, 0.5, 600,
0), (3, 10, 7, 11, 8, 70, 1.5, 1500, 0), (3, 10, 8, 12, 9, 100,
5, 6000, 0), (3, 10, 9, 13, 10, 150, 15, 15000, 0), (3, 10, 10,
14, 11, 200, 50, 60000, 1), (3, 10, 11, 15, 12, 300, 150,
150000, 3), (3, 10, 12, 16, 13, 500, 500, 600000, 10), (3, 10,
13, 17, 14, 700, 1500, 1500000, 30), (3, 10, 14, 18, 15, 1000,
```

```

5000, 6000000, 100), (3, 10, 15, 19, 16, 2000, 15000, 15000000,
300), (4, 30, 7, 9, 7, 50, 0.5, 1500, 0), (4, 30, 8, 11, 8, 70,
1.5, 6000, 0), (4, 30, 9, 12, 9, 100, 5, 15000, 0),
(4, 30, 10, 13, 10, 150, 15, 60000, 1), (4, 30, 11, 14, 11, 200,
50, 150000, 3), (4, 30, 12, 15, 12, 300, 150, 600000, 10),
(4, 30, 13, 16, 13, 500, 500, 1500000, 30), (4, 30, 14, 17, 14,
700, 1500, 6000000, 100), (4, 30, 15, 18, 15, 1000, 5000,
15000000, 300), (5, 1, 4, 10, 8, 50, 0, 150, 0), (5, 1, 5, 11, 9,
70, 0, 500, 0), (5, 1, 6, 12, 10, 100, 0, 1500, 0), (5, 1, 7, 13,
11, 150, 0, 5000, 0), (5, 1, 8, 14, 12, 200, 0, 15000, 0), (5,
1, 9, 15, 13, 300, 0, 50000, 0), (5, 1, 10, 16, 14, 500, 0,
150000, 1), (5, 1, 11, 17, 15, 700, 0, 150000, 3), (5, 1, 12,
18, 16, 1000, 0, 500000, 10), (5, 1, 13, 19, 17, 1500, 0,
1500000, 30), (5, 1, 14, 20, 18, 2000, 0, 5000000, 100), (5, 1,
15, 21, 19, 3000, 0, 15000000, 300)
GO
--insert star system
SET IDENTITY_INSERT star_systems ON
INSERT star_systems (id, name, star_mass) VALUES (1, 'Sol', 1)
SET IDENTITY_INSERT star_systems OFF
GO
SET IDENTITY_INSERT stellar_object_types ON
INSERT stellar_object_types (id, name)
VALUES (1, 'Terrestrial Planet'), (2, 'Gas Giant'), (3, 'Asteroid
Belt')
SET IDENTITY_INSERT stellar_object_types OFF
GO
SET IDENTITY_INSERT stellar_objects ON
INSERT stellar_objects (id, star_system, stellar_object_type,
name, orbital_radius, orbital_eccentricity, orbital_angle, mass,
planetary_radius, gravity, object_description)
VALUES (1, 1, 1, 'Mercury', 0.39, 0.21, 7, 0.055, 0.4937, 0.38,
'Very hot!'), (2, 1, 1, 'Venus', 0.71, 0.01, 3, 0.82, 0.4949,
0.91, 'Under pressure.'),
(3, 1, 1, 'Earth', 1, 0.02, 0, 1, 1, 1, 'We live here.'), (4, 1,
1, 'Mars', 1.5, 0.01, 2, 0.11, 0.5316, 0.378, 'Red planet.'),
(5, 1, 3, 'Main Belt', 2.7, 0, 0, 0, 0, 0, ''), (6, 1, 2,
'Jupiter', 5.2, 0.05, 1, 320, 11.2658, 2.528, ''), (7, 1, 2,
'Satur', 9.5, 0.06, 3, 95, 9.4937, 1.08, ''),
(8, 1, 2, 'Uranus', 19, 0.04, 1, 14, 4.0506, 0.886, ''), (9, 1,
2, 'Neptune', 30, 0.01, 2, 17, 3.9241, 1.14, '')
SET IDENTITY_INSERT stellar_objects OFF
GO
--insert spaceship
SET IDENTITY_INSERT spaceships ON

```

```

INSERT spaceships (id, name, tl, sm, streamlined, winged,
spin_gravity, stealth, dynamic_chameleon, automation, condition)
VALUES (1, 'Star Flower', 11, 8, 1, 0, 0, 0, 0, 1, 1)
SET IDENTITY_INSERT spaceships OFF
GO
SET IDENTITY_INSERT hull_slots ON
INSERT hull_slots (id, ship, section, location)
VALUES (1, 1, 1, 1), (2, 1, 1, 2), (3, 1, 1, 3), (4, 1, 1, 4),
(5, 1, 1, 5), (6, 1, 1, 6), (7, 1, 2, 1), (8, 1, 2, 2), (9, 1,
2, 3), (10, 1, 2, 4), (11, 1, 2, 5), (12, 1, 2, 6), (13, 1, 3,
1), (14, 1, 3, 2), (15, 1, 3, 3), (16, 1, 3, 4), (17, 1, 3, 5),
(18, 1, 3, 6), (19, 1, 1, 7), (20, 1, 3, 7)
SET IDENTITY_INSERT hull_slots OFF
GO
SET IDENTITY_INSERT habitats_to_slots ON
INSERT habitats_to_slots (id, slot, habitat, life_support)
VALUES (1, 8, 3, 1), (2, 9, 3, 1)
SET IDENTITY_INSERT habitats_to_slots OFF
GO
SET IDENTITY_INSERT rooms_to_habitats ON
INSERT rooms_to_habitats (id, habitat, room, amount)
VALUES (1, 1, 2, 6), (2, 2, 2, 4), (3, 2, 16, 2)
SET IDENTITY_INSERT rooms_to_habitats OFF
GO
SET IDENTITY_INSERT weapon_systems ON
INSERT weapon_systems (id, subtype, weapon_placement,
rate_of_fire, installed_weapons, configuration)
VALUES (1, 2, 1, 1, 1, 35)
SET IDENTITY_INSERT weapon_systems OFF
GO
SET IDENTITY_INSERT weapon_systems_to_slots ON
INSERT weapon_systems_to_slots (id, slot, weapon_system)
VALUES (1, 12, 1)
SET IDENTITY_INSERT weapon_systems_to_slots OFF
GO
SET IDENTITY_INSERT engines_to_slots ON
INSERT engines_to_slots (id, slot, engine, engine_variation,
high_thrust, ram_rocket)
VALUES (1, 14, 305, 65, 0, 0), (2, 15, 305, 65, 0, 0)
SET IDENTITY_INSERT engines_to_slots OFF
GO
SET IDENTITY_INSERT armor_to_slots ON
INSERT armor_to_slots (id, slot, armor_system, hardening)
VALUES (1, 1, 45, 0), (2, 7, 45, 0), (3, 13, 45, 0)
SET IDENTITY_INSERT armor_to_slots OFF
GO

```

```

SET IDENTITY_INSERT stardrive_engines_to_slots ON
INSERT stardrive_engines_to_slots (id, slot, stardrive_engine,
stardrive_engine_type, stardrive_engine_category)
VALUES (1, 16, 5, 1, 1), (2, 17, 5, 1, 1)
SET IDENTITY_INSERT stardrive_engines_to_slots OFF
GO
SET IDENTITY_INSERT sensor_arrays_to_slots ON
INSERT sensor_arrays_to_slots (id, slot, sensor_array,
sensor_array_type) VALUES (1, 6, 5, 1)
SET IDENTITY_INSERT sensor_arrays_to_slots OFF
GO
SET IDENTITY_INSERT cargo_holds_to_slots ON
INSERT cargo_holds_to_slots (id, slot, cargo_hold,
refrigerated_cargo, shielded_cargo)
VALUES (1, 2, 5, 0, 0), (2, 3, 5, 0, 0), (3, 4, 5, 0, 0), (4, 5,
5, 0, 0), (5, 10, 5, 0, 0), (6, 11, 5, 0, 0)
SET IDENTITY_INSERT cargo_holds_to_slots OFF
GO
SET IDENTITY_INSERT engine_rooms_to_slots ON
INSERT engine_rooms_to_slots (id, slot, engine_room) VALUES (1,
18, 4)
SET IDENTITY_INSERT engine_rooms_to_slots OFF
GO
SET IDENTITY_INSERT control_rooms_to_slots ON
INSERT control_rooms_to_slots (id, slot, control_room,
control_stations) VALUES (1, 19, 5, 4)
SET IDENTITY_INSERT control_rooms_to_slots OFF
GO
SET IDENTITY_INSERT power_plants_to_slots ON
INSERT power_plants_to_slots (id, slot, power_plant,
power_plant_variant, derated) VALUES (1, 20, 41, 12, 0)
SET IDENTITY_INSERT power_plants_to_slots OFF
GO
CREATE PROCEDURE delete_slots
@spaceship_id INT
AS
BEGIN
    DELETE FROM armor_to_slots
    WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
    DELETE FROM cargo_holds_to_slots
    WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
    DELETE FROM cloaking_devices_to_slots
    WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)

```

```

DELETE FROM contragravity_lifters_to_slots
WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
DELETE FROM control_rooms_to_slots
WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
DELETE FROM defensive_ecms_to_slots
WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
DELETE FROM engine_rooms_to_slots
WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
DELETE FROM engines_to_slots
WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
DELETE FROM external_clamps_to_slots
WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
DELETE FROM factories_to_slots
WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
DELETE FROM force_screens_to_slots
WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
DELETE FROM fuel_tanks_to_slots
WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
DELETE FROM rooms_to_habitats
WHERE habitat IN (SELECT id FROM habitats_to_slots WHERE
slot IN (SELECT id FROM hull_slots WHERE ship = @spaceship_id))
DELETE FROM habitats_to_slots
WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
DELETE FROM hangar_bays_to_slots
WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
DELETE FROM jump_gates_to_slots
WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
DELETE FROM mining_refinery_systems_to_slots
WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
DELETE FROM open_spaces_to_slots
WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
DELETE FROM passenger_seating_to_slots

```

```

        WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
        DELETE FROM power_plants_to_slots
        WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
        DELETE FROM ramscoops_to_slots
        WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
        DELETE FROM robot_arms_to_slots
        WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
        DELETE FROM sensor_arrays_to_slots
        WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
        DELETE FROM soft_landing_systems_to_slots
        WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
        DELETE FROM stardrive_engines_to_slots
        WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
        DELETE FROM stasis_webs_to_slots
        WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
        DELETE FROM weapon_systems
        WHERE id IN (SELECT weapon_system FROM
weapon_systems_to_slots WHERE slot IN (SELECT id FROM hull_slots
WHERE ship = @spaceship_id))
        DELETE FROM weapon_systems_to_slots
        WHERE slot IN (SELECT id FROM hull_slots WHERE ship =
@spaceship_id)
        DELETE FROM hull_slots
        WHERE ship = @spaceship_id
END
GO
CREATE PROCEDURE delete_spaceship
@spaceship_name VARCHAR(50)
AS
BEGIN
    DECLARE @spaceship_id INT;
    SET @spaceship_id = (SELECT id FROM spaceships WHERE name =
@spaceship_name);
    EXEC delete_slots @spaceship_id;
    DELETE FROM spaceships
    WHERE id = @spaceship_id
END

```