Правительство Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ИРКУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ИГУ)

УДК 519.685.8

СОГЛАСОВАНО
кто-то там
«Полюс-НТ»
_____ кто-то там
«____»_____ 2023 г.

УТВЕРЖДАЮ
кто-то там
где-то там
_____ кто-то там
«____»_____ 2023 г.

ОТЧЕТ
О ПРАКТИЧЕСКОЙ РАБОТЕ:

РАЗРАБОТКА ЭКСПЕРТНОЙ СИСТЕМЫ ПРИНЯТИЯ РЕШЕНИЙ ДЛЯ ИЭС

(заключительный)

Руководитель ПР,
где-то там,
кто-то там _____ кто-то там

Иркутск 2023

СПИСОК ИСПОЛНИТЕЛЕЙ

Исполнители:

Студент четвёртого курса “Прикладная информатика - Разработка ПО” факультета бизнес-коммуникаций и информатики ИГУ, 14423-ДБ	_____ подпись, дата	В. В. Шадрин (реферат, введение, разделы 1.1, 1.2, 1.3, 1.4, 1.5, 2.1)
Студент четвёртого курса “Прикладная информатика - Разработка ПО” факультета бизнес-коммуникаций и информатики ИГУ, 14423-ДБ	_____ подпись, дата	М.А. Клыпин (разделы 1.4, 1.5, 2.2, заключение)

РЕФЕРАТ

Отчёт 17 с., 0 кн., 7 рис., 0 табл., 0 источн., 0 прил.

ЭКСПЕРТНАЯ СИСТЕМА, ИЭС, АУКЦИОН, ПРИНЯТИЕ РЕШЕНИЙ, РЕКОМЕНДАЦИИ, PYTHON, PYQT

Объектом, вокруг которого велась практическая работа, является стенд “ИЭС”, предоставленный компанией “Полюс-НТ”.

Цель работы — разработка программного обеспечения, способного помогать пользователям стенда “ИЭС” принимать решения, ведущие к достижению победы в игре.

В процессе работы проводились экспериментальные игровые сессии на стенде “ИЭС”.

Продуктом практической работы является программа “Экспертная система принятия решений для ИЭС”, разработанная на языке программирования Python с использованием библиотек Pandas и PyQT6 и Numpy.

Основные показатели: высокий процент верно подобранных цен для аукциона, высокое соответствие действительности прогнозов на игру в рамках погрешности.

Степень внедрения — частичная, продукт не требует подключения к стенду и способен функционировать отдельно от него, однако не имеет ценности вне игр на стенде.

Эффективность программного обеспечения определяется отсутствием необходимости адаптирования стенда ИЭС под продукт, а также удобным и информативным интерфейсом.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На момент начала проведения проектной работы у компании “Полюс-НТ” не было готовой экспертной системы принятия решений для стенда “Интеллектуальные Энергетические системы” (далее “ИЭС”). При этом компания высказывала своё желание получить таковую для собственных целей. Из этого следует цель данной практической работы: проектирование и разработка успешной экспертной системы принятия решений для стенда “ИЭС” от компании “Полюс-НТ”. Для достижения данной цели должны быть выполнены следующие задачи:

1. анализ игры на стенде “ИЭС”;
2. проектирование экспертной системы принятия решений;
3. подбор инструментария для выполнения работы;
4. построение интерфейса будущего ПО;
5. реализация итогового продукта через написание кода.

При рассмотрении данного проекта с точки зрения участника игры на стенде “ИЭС” можно обнаружить высокий спрос на наличие экспертной системы принятия решений, связанный с чересчур высоким числом факторов, влияющих на игру. Такое число параметров не позволяет новому игроку быстро влиться в игровой процесс и вынуждает продолжительное время действовать наугад, допуская грубые ошибки. Вкупе с тем фактом, что игра на стенде подразумевает противостояние игроков друг другу, это образует очень высокий порог вхождения в игру, что влечёт за собой долгий период обучения и риск потери интереса у пользователей. Экспертная система принятия решений способна сгладить данную проблему.

Для разработчиков стенда ИЭС экспертная система принятия решений также может представлять прямую ценность, так как на её фундаменте возможно построение полноценного искусственного интеллекта для игры. ИИ ценен для игры тем, что способен упростить тестирование системы, а также создаёт возможность проведения игры в одиночном режиме (при участии всего одного живого игрока), чего в данный момент нет.

При построении экспертной системы принятия решений важно понимать цели, которые игра ставит перед игроком. Таковыми являются анализ графика погоды и построение на его основе стратегии будущей игры, успешная закупка необходимых объектов на аукционе по достойной цене, эффективное выстраивание объектов на игровом поле, а также написание алгоритма, способного выгодно распоряжаться предоставленными игроку ресурсами. Эти же цели должна преследовать и экспертная система.

ОСНОВНАЯ ЧАСТЬ ОТЧЁТА О ПР

1. Проектирование системы и подбор инструментария

1.1. Анализ игры, выделение фокуса проекта

Анализ игры показал, что наиболее значимыми для победы являются два аспекта игры: успешно проведённый аукцион и грамотное использование имеющейся энергии в дальнейшем. Поэтому в первую очередь был сделан упор на аукцион. Для его успешного проведения было решено построить математическую модель по высчитыванию рекомендованных цен на объекты. Были сделаны следующие утверждения:

- ценность объекта для игрока пропорционально его производству/потреблению;
- ценность объекта зависит от уже купленных противником таких же объектов;
- ценность объекта зависит от текущего соотношения между производством и потреблением энергии;
- ценность производимой энергии в начале игры выше, чем ценность производимой энергии в конце игры, так как в начале игры у нас нет накоплений.

По этим наблюдениям был собран набор из формул, каждая из которых возвращает значение от 0 до 1 для выбранного типа объектов. Ценность определённого типа объекта является средним арифметическим значением от данных формул. Чем выше ценность, тем менее выгодную цену нам нужно платить за данный объект, чтобы повысить наши шансы на победу в аукционе. Ценность также является значением от 0 до 1. Чтобы перевести это значение в настоящую цену, нужно умножить его на максимальную возможную цену типа объекта. Важно не забыть, что для потребителей значение ценности нужно инвертировать, так как там идёт игра на понижение, и чем меньше мы заплатим, тем хуже для нас.

1.2. О считывании предоставляемых данных

Особенностью предоставлаемых для анализа данных является то, что там представленны размеры потребляемой энергии для объектов “Микрорайон”, “Завод” и “Больница”, но вместо вырабатываемой энергии пишутся скорость ветра и яркость солнца. Для получения вырабатываемой энергии предоставленые значения нужно как-то обработать. Заведомо названы следующие условия:

- максимальная мощность объектов “СЭС” и “ВЭС” равна 15 МВт;
- “ВЭС” вырабатывает свою максимальную энергию при скорости ветра ⅔ от максимально возможной;
- количество энергии, вырабатываемое “ВЭС”, имеет прямую кубическую зависимость от скорости ветра;
- количество энергии, вырабатываемое “СЭС”, имеет прямую линейную зависимость от яркости солнца.

Также анализ предоставляемых для игры данных показал, что максимальные показатели скорости ветра и яркости солнца примерно равны 17 и 16 соответсвенно. На

основе всех вышеперечисленных сведений были выведены две формулы для получения энергии производителей. Для “ВЭС” используется следующая формула:

$$\text{энергия} = \left(\frac{x}{4.6}\right)^3,$$

где x является силой ветра в данный момент. Для “СЭС” же составлена следующая формула:

$$\text{энергия} = \frac{x}{1.07},$$

где x является яркостью солнца в данный момент. Было сделано допущение, что пользователь расставил свои производственные объекты в максимально эффективные позиции. При таком рассладе можно не беспокоиться о сильном отклонении действительных данных от прогнозируемых. Поэтому обе формулы могут и действительно применяются в момент считывания данных из файла, чтобы сразу же перевести показатели ветра и солнца в производственную энергию обектов “ВЭС” и “СЭС”.

1.3. Алгоритм нахождения рекомендуемых цен

Был составлен ряд факторов, влияющих на ценообразование. В него вошли:

- энергия, вырабатываемая или потребляемая объектом в течении игры, так как при разных прогнозах погоды разные типы объектов приносят наибольшую выгоду в течении игры;
- число объектов, доступных для покупки, так как общее число объектов ограничено и делится между всеми игроками, из-за чего можно легко оказаться в ситуации, когда пользователь не смог купить ни один объект определённого типа или даже категории;
- баланс между производимой и потребляемой энергией, так как при превосходстве одного над другим можно либо работать вхолостую, тратя производственные мощности в никуда, либо и вовсе не удовлетворять собственные потребности в энергии, из-за чего придётся докупать энергию извне по крайне невыгодной цене;
- показатели энергии у производителей в первые ходы игры, так как в самом начале у пользователя никак не могут иметься запасы энергии для борьбы нехваткой производства, из-за чего в случае низкой добычи энергии придётся докупать эту самую энергию извне за крайне невыгодную цену.

На основе данных факторов были созданы четыре параметра, также известных как “модификаторы”, как они называются внутри команды исполнителей. Все эти модификаторы представляют из себя числа от 0 до 1. Для рассчёта ценности типа объекта находится среднее арифметическое данных модификаторов. В дальнейшем найденная ценность, которая также является числом от 0 до 1, домножается на максимально возможный ценник типа объекта и таким образом трансформируется в итоговую цену.

Первый модификатор представляет из себя процент вырабатываемой или потребляемой за всю игру энергии объекта от всей вырабатываемой или потребляемой в

течении всей игры энергии. Для нахождения модификатора вычисляется отношение между всей энергией объекта за всю игру и всей энергией категории данного объекта. Как пример, отношение всей энергии объекта “Завод” ко всей энергии категории “Потребители” является первый модификатором для объекта “Завод”. Формула модификатора выглядит примерно так:

$$\text{модификатор } 1 = \frac{sum(\text{энергия типа объекта})}{sum(\text{энергия категории объекта})}.$$

Второй модификатор является процентом объектов одного типа, купленных противниками, от общего числа объектов данного типа в игре. Имеет следующую формулу:

$$\text{модификатор } 2 = \frac{\text{число(объекты противников[тип объекта])}}{\text{число(объекты игры[тип объекта])}}.$$

Третий модификатор отражает баланс между производимой и затрачиваемой энергией у объектов игрока. Для нахождения данного параметра сначала считаются сумма всей производимой игроком энергии в течении игры и сумма всей затрачиваемой игроком энергии в течении игры. Затем определяется какое из полученных значений больше, чтобы в дальнейшем вычислить отношение меньшего к большему. Полученное значение, лежащее в диапазоне 0-1, делится на два, после чего прибавляется и вычитается от 0.5, создавая ещё два значения. Из полученных двух значений большее является модификатором для объектов той категории, что имеет меньшую суммарную энергию, в то время как меньшее значение будет модификатором для категории с большей суммарной энергией. Благодаря этому объекты обеих категорий получают модификаторы, но те, что находятся в проигрыше, имеют больший приоритет, который растёт по мере роста их разрыва в суммах. Формулы для данного модификатора выглядит примерно так:

$$\text{разница} = \frac{\text{сумма энергии потребителей}}{\text{сумма энергии производителей}}$$

или

$$\text{разница} = \frac{\text{сумма энергии производителей}}{\text{сумма энергии потребителей}}$$

в зависимости от того, у кого сумма больше (числитель < знаменатель);

$$\text{модификатор } 3.1 = 0.5 + \frac{\text{разница}}{2};$$

$$\text{модификатор } 3.2 = 0.5 - \frac{\text{разница}}{2}.$$

В том случае, если одна из категорий в сумме энергии содержит нулевое значение, данной категории присваивается модификатор со значением 1, в то время как второй категории присваивается модификатор со значением 0. Если же суммы энергии обеих категорий равны нулю, то обеим категориям присваиваются модификаторы со значением 0.5.

Четвертый модификатор рассчитан только на производителей энергии и отражает какой из типов объектов приносит большую энергию в первые ходы игры (конкретно в

первые 25% ходов от всей продолжительности игры). Для высчитывания данного модификатора берётся сумма энергии объекта за определённое число ходов, а затем делится на общее число производимой энергии за то же число ходов. Формула модификатора выглядит примерно так:

$$\text{модификатор 4} = \frac{sum(\text{энергия типа объекта}[n \text{ ходов}])}{sum(\text{энергия категории объекта}[n \text{ ходов}])}.$$

Итоговая цена для производителей рассчитывается так:

$$\text{цена производителя} = 20 * \frac{\text{модификатор 1}+\text{модификатор 2}+\text{модификатор 3}+\text{модификатор 4}}{4}.$$

При расчёте цены потребителя нужно инвертировать полученную ценность, так как, в отличии от объектов-производителей, объекты-потребители с ростом цены несут большую пользу для игроков, из-за чего на аукционе все пытаются задать данным объектам цену повыше. Поэтому итоговая цена для потребителей рассчитывается так:

$$\text{цена потребителя} = 10 * (1 - \frac{\text{модификатор 1}+\text{модификатор 2}+\text{модификатор 3}}{3}).$$

1.4. Вычисление вспомогательных для пользователя данных

Для контроля соотношения потребителей и производителей была придумана метрика “Среднее накопление энергии”. Она является разницей между суммой средней энергии за ход производителей, что находятся под контролем игрока, и суммой средней энергии за ход потребителей, что также находятся под контролем игрока. Полученное значение плохо характеризует баланс энергии игрока в каждый отдельный момент игры, но даёт базовое понимание этого баланса, на основе чего игрок может предполагать какой конкретно категории объектов ему не хватает.

Очень важно учитывать потери энергии при её транспортировке. Согласно инструкции, максимальный уровень потерь в 20% достигается при суммарной мощности на ветке в 18 МВт независимо от того, генератор или потребитель находятся на этой ветке. Так как расстановка объектов игроками является крайне субъективным делом, то расчет потерь энергии был осуществлён в довольно грубой форме. Было сделано допущение, что максимальное количество веток составляет пять при использовании одной миниподстанции. Две из них проложены от производителей до подстанции, ещё две от потребителей до миниподстанции, а последняя соединяет подстанцию и миниподстанцию. Если так рассуждать, то можно предположить при каких показателях энергии на объектах будут какие потери на этих ветках. Так, к примеру, так как у нас есть две ветки под производителей, то мы можем предположить, что при суммарной энергии в 25 МВт 18 из них будут протекать по одной ветке, а оставшиеся 7 по другой, из-за чего на одной ветке потери будут максимальными, а на другой относительными. Для расчёта относительных потерь была предпринята попытка вывести формулу, которая используется стендом для данных расчётов. Зная из инструкции, что данная формула является квадратичной, была предположена следующая запись:

$$\text{процент потерь} = (\frac{x}{4})^{2}/100,$$

где x является текущей мощностью на ветке. Если возвращаться к примеру, описанному выше, то это значит, что у нас будут потери в 3,6 МВт от 18 МВт, а также 0,2 МВт от 7 МВт. Таким образом и считаются ожидаемые потери на ветках, что является несомненно оптимистичным и крайне неточным, но всё же результатом, на основе которого можно иметь хоть какое-то представление о потерях энергии во время игры.

1.5. Подбор инструментария

Для разработки приложения был выбран язык программирования Python из-за следующих преимуществ.

1. Кроссплатформенность: Python поддерживает различные операционные системы, такие как Windows, macOS и Linux, что позволяет создавать приложения, которые могут работать на разных платформах без изменений в исходном коде.

2. Большое сообщество: Python имеет огромное сообщество разработчиков, что означает, что всегда можно найти поддержку, документацию и готовые решения для различных задач.

3. Множество библиотек и фреймворков: Python имеет множество библиотек и фреймворков, которые упрощают разработку desktop приложений, такие как PyQt, Tkinter, Kivy и другие.

4. Простота в использовании: Python позволяет быстро создавать прототипы и разрабатывать приложения благодаря своему простому и понятному синтаксису.

5. Расширяемость: Python легко интегрируется с другими языками программирования, такими как C/C++, что позволяет использовать существующий код и библиотеки для ускорения разработки.

В качестве среды разработки был выбран Visual Studio Code, как бесплатное, современное решение.

Для построения интерфейса к Python была подключена библиотека PyQt6. Она содержит в себе набор расширений графического фреймворка Qt, выполненный в виде библиотеки для языка программирования Python. PyQt6 удобен в использовании, лёгок в освоении, предоставляет широкий набор функций и классов, а также абсолютно бесплатен при работе над некоммерческим проектом. Всё это ускоряет разработку и позволяет сконцентрироваться на функционале программы, а не на её внешнем виде. Так как проект не рассматривается как коммерческий, то и проблем с лицензией не наблюдается.

Сильная сторона PyQt состоит в том, что это ответвление от крупного фреймворка Qt, благодаря чему библиотека обладает мультиплатформенностью, а помимо самой библиотеки существуют и отдельные приложения, упрощающие работу с ней.

Qt Designer — кроссплатформенная свободная среда для разработки графических интерфейсов для программ, использующих библиотеку Qt. Входит в состав Qt framework. Данная среда является наиболее удобным способом конструировать графические интерфейсы на основе фреймворка Qt, которые после можно импортировать в файл с

расширением “py” (файл языка программирования Python), что является высокой экономией времени в сравнении с ручной прописью всех элементов интерфейса. Так как в проекте используется библиотека PyQt6, являющаяся частью фреймворка Qt, Qt Designer – необходимый инструмент.

Заведомо известно, что программа будет взаимодействовать со множеством числовых данных в формате “csv”, а также потребует отображение графиков этих данных. Поэтому к проекту были подключены библиотеки Pandas и matplotlib. Pandas предоставляет контейнер DataFrame, позволяющий в матричном виде удобно хранить и обрабатывать большие массивы данных. Для ещё большего контроля данных используется библиотека NumPy, нацеленная на действия с массивами. Для отображения этих данных в красивом графическом виде из библиотеки matplotlib взят модуль pyplot, строящий графики и предоставляющий инструменты для взаимодействия с ними.

2. Создание ПО: интерфейс и логика

2.1. Интерфейс приложения

Для написания интерфейса используется библиотека PyQt6. Дизайн всех окон был разработан в приложении Qt Designer и импортирован в файл “py” через консольную команду “pyuic6 -o output.py -x input.ui”, где “input.ui” — это файл из Qt Designer, а “output.py” — наименование файла с кодом интерфейса, который будет создан. Всего было создано три файла интерфейса под три класса: “qt_main.py” с “MyMainWindow”, “qt_add_object.py” с “DialogAddObject” и “qt_new_objects.py” с “DialogNewObjects”. Каждый из данных классов является окном, из которых состоит интерфейс программы.

Главное окно, являющееся объектом класса “MyMainWindow” (рис. 1), содержит в себе две области: область под управление графиком (слева) и область “Аукцион” для взаимодействий с объектами игры (справа).

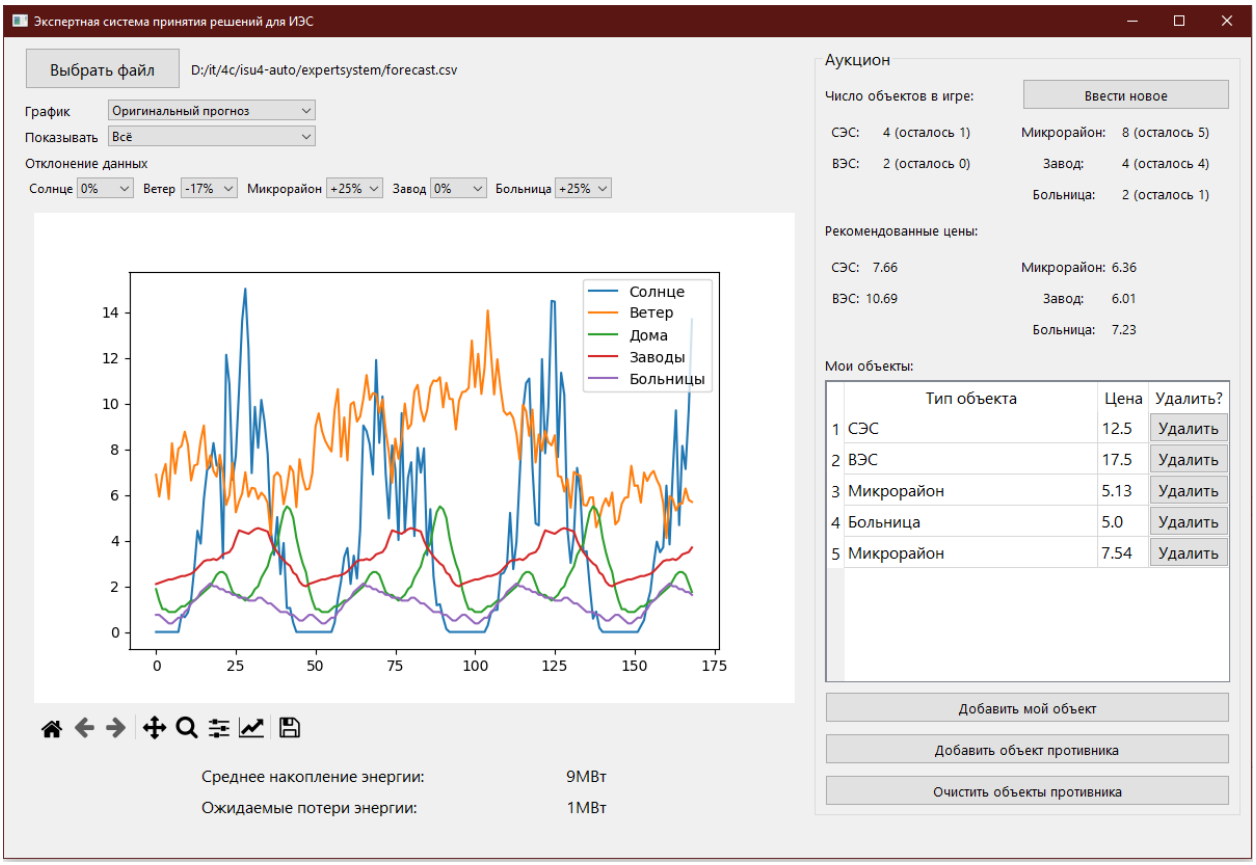

Рисунок 1. Главное окно, объект класса “MyMainWindow”

Область с графиком состоит из самого графика, нескольких выпадающих списков для его контроля, вспомогательных элементов, выводящих полезную игроку информацию (“Среднее накопление энергии” и “Ожидаемые потери энергии”), и кнопки выбора файла для загрузки данных в программу. Самым первым элементом является кнопка, которая открывает “Проводник” операционной системы и предлагает выбрать файл с расширением “csv”. После выбора файла его полное наименование отобразиться справа от кнопки. Ниже расположены выпадающие списки “График” и “Показывать”, а также целый набор из списков под общим названием “Отклонение данных”. “График” позволяет выбрать пользователю какие именно данные будут отображаться: оригинальные из файла и собранные на основе объектов, что находятся в собственности игрока. “Показывать” отвечает за фильтрацию данных на графике: можно показывать как сразу все объекты, так и только потребителей или только производителей. Списки из группы “Отклонение данных” добавляют в графики погрешности, возможные в игре: -17%, +25% или без изменений. Ниже находится участок для отрисовки самого графика. В этом же участке присутствует управляющий инструментарий, предоставляемый модулем pyplot. Данный инструментарий позволяет менять элементы графика: легенда, цвета, оси, область видимости и т.д. В самом низу экрана расположены поля “Среднее накопление энергии” и “Ожидаемые потери энергии”, напротив которых выводятся соответствующие значения.

Область “Аукцион” сама по себе делится на три области: данные о количестве участвующих в игре объектов, рекомендуемые цены на объекты и таблица с объектами игрока.

Данные о количестве участвующих в игре объектов сообщают не только общее число каждого типа, но и сколько таких объектов осталось неиспользованными (то есть они не куплены ни игроком, ни его противниками). Данная информация обновляется при каждом добавлении или удалении объекта из владения игроков. Тут же находится кнопка “Ввести новое”, при нажатии на которую открывается диалоговое окно “Ввод объектов игры” класса “DialogNewObjecst” (рис. 2). Через данное окно пользователь вводит информацию о количестве объектов каждого типа в игре. Каждый раз, когда пользователь вводит новые данные, массив с объектами противника обнуляется, так что нужно быть внимательным и следить за корректным вводом информации с первого раза.

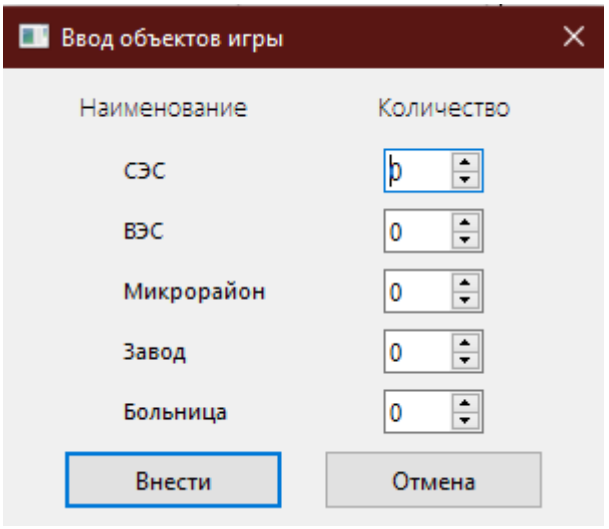

Рисунок 2. Диалоговое окно, объект класса “DialogNewObjects”

Участок с выводом рекомендуемых цен оформлен крайне просто: по аналогии с предыдущей областью показывается пять полей под каждый тип объектов, напротив которых пишутся предлагаемые цены. Данные поля впервые заполняются после выбора пользователем файла с данными, а также ввода участвующих в игре объектов. После этого данные поля пересчитываются автоматически после каждой покупки или при удалении уже купленных объектов.

Таблица с объектами пользователя состоит из трёх колонок: “Тип объекта”, “Цена” и “Удалить?”. В первых двух столбцах отображаются соответствующие их названию данные, а в ячейках третьего столбца создаются кнопки “Удалить”, по нажатию на которые объект, на чьей строке была нажата кнопка, удаляется из таблицы и из соответствующего массива. Также стоит отметить, что данная таблица вложена в пролистываемую область, из-за чего при полном заполнении таблицы объектами появляется полоса прокрутки, позволяющая продолжать заполнять таблицу и дальше. Ниже пролистываемого участка области с таблицей созданы три кнопки: “Добавить мой объект”, “Добавить объект противника” и “Очистить объекты противника”. Первые две кнопки вызывают окно “Ввод купленного объекта” класса “DialogAddObject” (рис. 3), через которое выбираются тип и цена купленных объектов. Если окно было вызвано через “Добавить мой объект”, то полученная информация добавляется в таблицу объектов и в массив с объектами игрока. Если же использовалась кнопка “Добавить объект противника”, то полученная

информация вносится только в массив объектов других игроков. Кнопка “Очистить объекты противника” этот самый массив обнуляет.

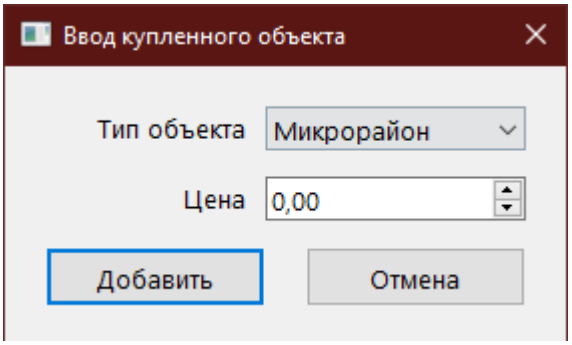

Рисунок 3. Диалоговое окно, объект класса “DialogAddObject”

2.2. Обработка данных

Главным классом для обработки данных является класс DataProcessor. Этот класс хранит в себе данные об игровых объектах, их тарифах и необходимую для внутренних функций приложения информацию.

Также были созданы следующие вспомогательные классы и TypeAlias’ы:

- MostValuedConsumer и MostValuedProducer - классы, хранящие в себе тип объекта, который добывает (или потребляет) больше всего энергии исходя из прогнозов (поле name), и количество добычи (потребления) в поле value;
- ProducerName - TypeAlias для типа Literal['Ветер', 'Солнце'];
- ConsumerName - TypeAlias для типа Literal['Больницы', 'Дома', 'Заводы']
- GameObject - класс, состоящий из поля object_type - тип объекта - и tariff - тариф данного объекта.

Функции класса DataProcessor:

- parse(path: str) - принимает путь до csv файла, считывает данные в виде DataFrame, переводит показатели ветра и солнца в энергию “ВЭС” и “СЭС”, а также просчитывает погрешности (+25%, -17%) для каждого объекта;
- prepare_data(data: DataFrame) - функция, которая подсчитывает сумму производства и сумму потребления всех объектов;
- getMostValued() - функция, которая находит объекты с самым высоким потреблением и объект с самым высоким производством;
- add_object(object: GameObject, enemy: bool) - функция для добавления игрового объекта, флаг enemy отвечает куда будет добавлен объект - к объектам противника или к объектом пользователя. Если добавление произошло к объектам пользователя, то будет пересчитано СНЭ (среднее накопление энергии);
- get_mean_energy_store() - функция, которая высчитывает среднее накопление энергии;
- get_data_for_graphics() - функция, которая возвращает словарь, содержащий текущее потребление или производство по типу объекта.

- get_values() - функция, которая возвращает рекомендации по ценам на каждый тип объекта. По ходу выполнения функции вычисляются четыре модификатора цены:
    - первый модификатор является отношением между всей энергией объекта к сумме всей энергии категории данной энергии (отношение всей энергии объекта “Завод” ко всей энергии категории “Потребители”, к примеру);
    - второй модификатор рассчитывается как отношение между количеством купленных объектов данного типа противником и общим количеством данных объектов в игре;
    - третий модификатор рассчитывается как отношение средней получаемой игроком энергии к средней затраченной игроком энергии;
    - четвертый модификатор рассчитывается только для производителей, и он определяет наилучшего производителя в первые 25% ходов, что важно для успешного старта игры, так как в начале игры у пользователей нет каких-либо накоплений энергии и легко случайно попасть в ситуацию, когда придётся закупать энергию извне по крайне невыгодной цене.

Итоговая цена считается по формуле, изображенной на рисунке 4.

```
# Определение ценности
prices["Солнце"] = (mods[0][0]+mods[0][1]+mods[0][2]+mods[0][3]) / 4
prices["Ветер"] = (mods[1][0]+mods[1][1]+mods[1][2]+mods[1][3]) / 4
prices["Дома"] = (mods[2][0]+mods[2][1]+mods[2][2]) / 3
prices["Заводы"] = (mods[3][0]+mods[3][1]+mods[3][2]) / 3
prices["Больницы"] = (mods[4][0]+mods[4][1]+mods[4][2]) / 3

# Определение цены
prices["Солнце"] = 20*prices["Солнце"]
prices["Ветер"] = 20*prices["Ветер"]
prices["Дома"] = 10*(1-prices["Дома"])
prices["Заводы"] = 10*(1-prices["Заводы"])
prices["Больницы"] = 10*(1-prices["Больницы"])

return prices
```

Рисунок 4. Расчет итоговой стоимости на основе модификаторов

Учет потерь на линиях происходит в функции update_mean_labels. Расчет делится на несколько этапов:

- расчет потерь на линии от производителя до главной станции (Рисунок 5);
- расчёт потери энергии при передаче от главной станции до миниподстанции (Рисунок 6);
- расчёт потери энергии при передаче от миниподстанции до потребителей изображен на рисунке 7.

```
mean_lost_plus = np.mean(self.data_processor.data["Солнце"])*self.my_objects_count["СЭС"]+np.mean(self.data_processor.data["Ветер"])*self.my_objects_count["ВЭС"]
# Рассчёт примерной потери энергии при передаче от производителей до главной станции
if mean_lost_plus > 36:
    mean_lost_plus *= 0.2
elif mean_lost_plus > 18:
    mean_lost_plus = 18*0.2 + (mean_lost_plus-18)*(((mean_lost_plus-18)/4)**2)/100
else:
    mean_lost_plus = mean_lost_plus * ((mean_lost_plus/4)**2) / 100
```

Рисунок 5. Расчет потерь на линии от производителя до главной станции.

```
# Рассчёт потери энергии при передаче от миниподстанции до потребителей
if mean_lost_min > 36:
    mean_lost_min *= 0.2
elif mean_lost_min > 18:
    mean_lost_min = 18*0.2 + (mean_lost_min-18)*(((mean_lost_min-18)/4)**2)/100
else:
    mean_lost_min = mean_lost_min * ((mean_lost_min/4)**2) / 100
mean_lost_min += mean_lost_min2
```

Рисунок 6. Расчет потери энергии при передаче от миниподстанции до потребителей.

```
# Рассчёт потери энергии при передаче от главной станции до миниподстанции
mean_lost_min2 = 0
if mean_lost_min > 18:
    mean_lost_min2 = mean_lost_min*0.2
    mean_lost_min = mean_lost_min*0.8
```

Рисунок 7. Расчет потери энергии при передаче от главной станции до минподстанции.

ЗАКЛЮЧЕНИЕ

В результате работы было разработано приложение, способное помогать пользователям стенда “ИЭС” принимать решения, ведущие к достижению победы в игре. Приложение поддерживает добавление игровых объектов, а также их удаление, показывает графики на основе текущей игровой ситуации и получаемых из csv-файла прогнозов. Но во время разработки были выявлены возможности для улучшения приложения, также были выявлены недостатки, которые мы не смогли спрогнозировать на стадии проектирования.