ИЗУЧЕНИЕ ОСНОВ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ НА ПРИМЕРЕ РАЗРАБОТКИ ИНКРЕМЕНТАЛЬНОЙ ИГРЫ

Крохалев К.О., студент,

Тазетдинов Б.И., к.ф.-м.н.,

Тазетдинова Ю.А., к.ф.-м.н.,

Бирский филиал УУНиТ, г. Бирск, Россия

Аннотация. В статье рассматриваются основные принципы объектно-ориентированного программирования (ООП) как одной из ключевых парадигм современного программирования. Показано, как с помощью классов и экземпляров можно моделировать реальные и абстрактные объекты. Для практической иллюстрации подхода представлен пример разработки простого прототипа инкрементальной игры, демонстрирующий применение базовых концепций ООП, а также их влияние на удобство разработки, масштабируемость и поддерживаемость программного кода.

Ключевые слова: класс, объект, инкапсуляция, наследование, полиморфизм, агрегация, метод.

Первым шагом в организации программы под ООП является выделение сущностей, которые будут представлены классами. В нашем проекте ключевыми классами стали:

· **Player (Игрок)**— объект, который управляется пользователем, обладает позицией, размерами, ресурсами (деньгами, маной) и методами для перемещения.

· **Ore (Руда)**— ресурс на игровом поле, с координатами, ценностью и визуальными характеристиками, который игрок собирает.

Выделение этих классов отражает суть инкрементальной игры — игрок собирает ресурсы для накопления богатства, а классы помогают логично разделить функциональность.

Каждый класс – это своего рода “чёрный ящик” с характеристиками и действиями. Такие основные концепции ООП, как инкапсуляция и абстракция, воплощаются именно здесь.

Пример класса Player:

· **Свойства:** X,Y— координаты на экране,Size— размер, Money— текущие деньги, Mana— ману, Speed— скорость перемещения.

· **Методы:** Move(dx, dy) изменяет позицию игрока с учётом нарисованного пространства. Позволяет двигаться в разные стороны.

Пример класса Ore:

· **Свойства:**X,Y— позиция, Size— размер для отрисовки,Value— сколько денег получит игрок при сборе, OreColor— цвет для визуализации.

· **Методы:** Конструктор для инициализации свойств, отдельные действия пока не реализованы, но логика сбора находится во внешнем управляющем классе (форме).

Это разделение позволяет поддерживать чёткую структуру и держать код гибким к изменениям.

Очень важный момент — классы управляют внутренним состоянием и предоставляют методы для доступа и изменения данных.

В проекте взаимодействие происходит так:

· В классе формы (Form1) создан список объектов ores(руда).

· При событии клавиатурного ввода (KeyDown) вызывается метод player.Move(dx, dy), меняющий позицию игрока.

· После перемещения проверяется столкновение игрока с рудами. Если позиция игрока близка к позиции руды (расстояние меньше размеров), то руда удаляется из списка, а деньги игрока увеличиваются на значение ore.Value.

· Для отрисовки используются свойства каждой сущности — в методе OnPaint циклом рисуются игрок и все объекты руды, используя Graphics.FillRectangleиDrawString.

Этот поток показывает чёткое разделение ответственности между объектами: игрок отвечает за своё состояние и движение, руда — за свои параметры, а форма — за логику взаимодействия и отрисовку.

Обработка ввода и управление состояниями

Для удобства пользователя реализована возможность одновременно нажимать несколько клавиш (например, движение по диагонали):

· В классе формы сохранён набор нажатых клавишHashSet<Keys> pressedKeys.

· При нажатии клавиши она добавляется в набор, при отпускании — удаляется.

· В обработчике нажатий проверяется состояние всех нажатых кнопок, и на их основе вычисляется направление движения.

Такой подход показывает работу с коллекциями и событиями, что важно в современных играх.

Масштабируемость и расширяемость

Проект изначально строится с расчётом на расширение функционала:

· Впереди добавление отдельных классов для питомцев (Pet), с собственными методами поведения.

· Возможность внедрить автоматическую генерацию руды, систему апгрейдов, эффекты, интерфейс и прочие игровые механики.

· Использование ресурсов (картинок) для визуального улучшения игры.

При адекватной структуре кода новые возможности добавляются без больших изменений существующей логики, что делает ООП важным инструментом при разработке. При работе над проектом использовались следующие источники информации [1 - 3].

Таким образом, изучение ООП на примере инкрементальной игры позволяет понять фундаментальные понятия — классы, объекты, инкапсуляция, взаимодействие — в контексте конкретной задачки. Такой подход облегчает не только обучение, но и помогает быстро создавать масштабируемые и читаемые программы с понятной архитектурой.

Литература

1. Официальный сайт Microsoft// URL:https :// learn . microsoft . com / ru - ru / dotnet / cs harp / fundamentals / tutorials / oop (дата обращения: 15.10.2025).

2. Тазетдинов Б.И., Тазетдинова Ю.А. Объектно-ориентированное программирование на языке C#. Часть 1. Учебное пособие для ВУЗов. Бирск: Изд-во Бирского филиала БашГУ, 2021 – 183 с.

3. Тазетдинов Б.И., Тазетдинова Ю.А. Объектно-ориентированное программирование на языке C#. Часть 2. учебное пособие для ВУЗов.
Бирск: Изд-во Бирского филиала БашГУ, 2021 – 204 с.