# ВВЕДЕНИЕ

Современные компьютерные технологии предоставляют множество возможностей для создания интерактивных приложений, таких как игры, которые одновременно развлекают и способствуют развитию навыков программирования, алгоритмизации и работы с графикой. Разработка игровых приложений сохраняет свою актуальность благодаря востребованности в сфере образования и индустрии развлечений. Особое внимание привлекают игры, включающие элементы стратегии и взаимодействия между игроками, поскольку они помогают развивать логическое мышление и умение принимать решения в динамичных условиях.

Эта курсовая работа сосредоточена на создании игрового приложения «Кольцевые гонки» для платформы *Windows Forms* с применением графики *OpenGL*. Главная цель – разработать полноценное приложение для двух игроков, где реализована механика гонок на одном экране с появлением призов на трассе. В процессе работы решаются задачи, связанные с созданием алгоритмов для управления автомобилями и взаимодействия объектов, использованием *OpenGL* для обеспечения качественной графики, применением шаблонов проектирования для гибкости кода, а также тестированием и проверкой работоспособности приложения.

Для реализации используются актуальные технологии и инструменты. Язык *C*# в среде *Windows Forms* упрощает разработку интерфейса и логики, а библиотека *OpenGL* обеспечивает плавное и производительное отображение игрового процесса. Применение таких шаблонов проектирования, как «фабричный метод» и «декоратор», делает код модульным и легко расширяемым, что соответствует современным подходам к разработке программного обеспечения.

Созданное приложение может стать полезным примером для изучения основ разработки игр и базой для дальнейших улучшений. В рамках работы анализируются существующие подходы, разрабатывается алгоритмическая основа, реализуется программная часть и проводится тестирование, что позволяет оценить эффективность предложенных решений.

**2 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ИГРОВОГО ПРИЛОЖЕНИЯ   
 «КОЛЬЦЕВЫЕ ГОНКИ»**

**2.1 Программные средства и шаблоны проектирования «Фабричный   
 метод» и «Декоратор»**

Для разработки программного обеспечения используется интегрированная среда *Visual* *Studio* 2022, а также компоненты, обеспечивающие поддержку платформы .*NET* и позволяющие создавать как библиотеки классов, так и клиентские приложения на базе *Windows* *Forms*.

Для интеграции с *OpenGL* подключаются необходимые библиотеки: *OpenTK* и *OpenTK*.*GLConftrol*.

Для создания спрайтовой графики в формате .*png* используется графический редактор *Aseprite*.

С целью обеспечения расширяемости и гибкости архитектуры разрабатываемого приложения применяются шаблоны проектирования.

Фабричный метод (*Factory* *Method*) – это один из наиболее популярных порождающих шаблонов проектирования. Его основная задача – делегировать создание объектов подклассам, тем самым позволяя использовать в коде объекты, не зная их конкретных классов. Это особенно важно в ситуациях, когда система должна быть расширяема и модифицируема без изменения существующего кода.

Идея шаблона заключается в создании абстрактного метода (или интерфейса), который определяет общий способ создания объектов. Конкретные реализации этого метода в подклассах возвращают объекты конкретных типов. Таким образом, клиентский код работает с абстракцией, а конкретный тип объекта подсовывается фабрикой.

В разрабатываемом приложении «Кольцевые гонки» используется шаблон проектирования Фабричный метод (*Factory* *Method*) для создания различных типов призов (топливо, ускорение, замедление) без привязки к конкретным классам в клиентском коде. Это позволяет изолировать процесс создания объектов от их использования и облегчает добавление новых типов призов в будущем. Этот подход позволяет создавать призы динамически и гибко управлять их типами, не нарушая принцип открытости/закрытости. Например, чтобы добавить новый тип приза, достаточно реализовать новый класс-награду и соответствующую фабрику, не изменяя остальной код.

На рисунке 2.1 представлена схема реализаии паттерна «фабричный метод».

В приложении «Кольцевые гонки» с помощью шаблона проектирования «Декоратор» (*Decorator*) происходит изменение максимальной скорости машины при подборе приза. Основной класс *Car* содержит ссылку на текущий активный декоратор (*\_currentDecorator*), который может динамически изменять поведение машины, не изменяя её структуру.

Каждый конкретный эффект, такой как ускорение или замедление, наследует абстрактный класс *CarDecorator*, который определяет общий интерфейс для всех декораторов. При активации эффекта машина вызывает метод, который устанавливает новый декоратор, применяющий соответствующее изменение – например, изменение параметра *ForwardMaxSpeed* у конфигурации движения машины. Через заданный интервал времени декоратор автоматически завершает своё действие и возвращает параметры в исходное состояние.

Таким образом, декоратор оборачивает машину дополнительным временным поведением без изменения самого класса *Car*. Это позволяет гибко расширять поведение автомобиля (например, добавлять новые эффекты) и применять их независимо друг от друга, что является типичным применением шаблона Декоратор.

**2.2 Структура классов приложения «Кольцевые гонки»**

***2.2.1*** Разрабатываемое приложение «Кольцевые гонки» представляет собой двухмерную гоночную игру для двух игроков, разработанную на платформе *C*# с использованием *Windows* *Forms* для создания пользовательского интерфейса и библиотеки *OpenTK* для рендеринга графики посредством *OpenGL*. Архитектура проекта разделяет логику представления (*UI*), основную игровую механику и вспомогательные утилиты по разным пространствам имен (*RingRaceApp*, *RingRaceLab*, *RingRaceLab*.*Game*, *RingRaceLab*.*Menu*), обеспечивая модульность и управляемость кода.

Основным окном приложения является класс *Form*1 (Приложение А, код программы *Form*1.*cs*), расположенный в пространстве имен *RingRaceApp*. Ключевая роль *Form*1 заключается в инициализации и управлении двумя основными состояниями приложения: главным меню и непосредственно игровым процессом. Это достигается через создание экземпляров контроллеров, реализующих интерфейсы *IMenuController* и *IGameController*. Форма хранит ссылки на панели (*MenuPanel* и *GamePanel*), связанные с каждым контроллером, и добавляет их в коллекцию своих элементов управления (*Controls*). Переключение между меню и игрой осуществляется методами *ShowMenu*() и *ShowGame*(), которые отвечают за скрытие одной панели и отображение другой. Форма также устанавливает *KeyPreview* = *true*, что позволяет ей перехватывать события клавиатуры до того, как они достигнут дочерних элементов управления, что важно для глобальных обработчиков, например, выхода из игры по нажатию *Escape*.

Когда игрок выбирает старт гонки в меню, управление переходит к *GameController*, который, в свою очередь, создает и запускает *GameManager* (Приложение А, код программы *GameManager*.*cs*). *GameManager*, находящийся в *RingRaceLab*, является сердцем игрового процесса. Он отвечает за координацию всех аспектов активной гонки. В его конструктор передаются все необходимые данные для инициализации сессии: информация о выбранной трассе (текстура и карта коллизий), стартовые позиции машин, координаты финишной линии, текстуры для автомобилей обоих игроков и размеры игрового окна. Основные обязанности *GameManager*:

должны реализовать все его наследники (например, *Track* и *Car*);