

ОТЧЕТ

по практическому занятию №1

\1

1. Титульный лист (данные для оформления)

? ФИО студента: \1

? Группа: \1

? Название работы: \1

Примечание: титульный лист обычно оформляется по шаблону кафедры/вуза. Если есть официальный шаблон, перенести данные из этого раздела в шаблон.

2. Цель работы

Создать минимальный рабочий каркас игрового движка с поддержкой игрового цикла и визуализации графического вывода через адаптер рендеринга.

3. Постановка задачи (по заданию из PDF)

Требовалось реализовать:

1. Класс \1, управляющий жизненным циклом приложения.
2. Систему состояний игры (например, загрузка, меню, игровой режим).
3. Расчет \1 для корректного обновления логики между кадрами.
4. Систему логирования событий и ошибок.
5. Адаптер рендеринга (\1) для отделения основной логики от графической библиотеки.
6. Отрисовку тестового примитива в графическом окне.

Дополнительное задание:

1. Обработка клавиатуры (стрелки) для перемещения объекта.
2. Обработка мыши:
 - ? ЛКМ — масштабирование объекта,
 - ? ПКМ — вращение объекта.
3. Плавное изменение параметров с учетом \1.

4. Используемые средства

? Язык программирования: \1

? Система сборки: \1

? Графическая библиотека: \1 (через \1)

? Среда разработки: \1

? Контроль версий: \1

? Репозиторий проекта: \1

5. Общая архитектура решения

Проект реализован как минимальный каркас игрового приложения с разделением ответственности между модулями.

5.1. Центральный класс \1

Класс \1 управляет жизненным циклом программы:

- ? инициализация логирования и рендера;
- ? запуск главного цикла;
- ? вычисление \1 с помощью \1;
- ? обработка переходов между состояниями;
- ? завершение работы и освобождение ресурсов.

Основной цикл организован по схеме:

1. вычисление \1;
2. чтение ввода;
3. обновление текущего состояния;
4. отрисовка кадра.

5.2. Система состояний игры

Реализована базовая система состояний с общим интерфейсом:

- ? \1
- ? \1
- ? \1

Каждое состояние имеет методы входа, выхода, обновления и отрисовки. Это позволяет изолировать логику разных режимов работы приложения и упростить расширение проекта.

5.3. Адаптер рендеринга (\1)

Рендеринг вынесен за интерфейс \1, который скрывает детали конкретной библиотеки.

В проекте реализован \1, который обеспечивает:

- ? создание окна;
- ? обработку ввода;
- ? начало/завершение кадра;
- ? отрисовку примитива;

- ? вывод текста;
- ? корректное завершение работы окна.

Такой подход позволяет заменить графическую библиотеку без изменения основного каркаса приложения.

5.4. Система логирования

Реализован модуль \1, который пишет сообщения:

- ? в консоль;
- ? в файл \1.

Логируются:

- ? запуск и завершение приложения;
- ? переходы между состояниями;
- ? значения \1.

5.5. Обработка ввода

Обработка управления вынесена в класс \1.

Реализовано:

- ? перемещение примитива стрелками;
- ? масштабирование по ЛКМ;
- ? вращение по ПКМ;
- ? плавное изменение параметров с учетом \1.

6. Структура проекта

Основные файлы проекта:

- ? \1 — конфигурация сборки
- ? \1 — точка входа
- ? \1, \1 — каркас приложения и главный цикл
- ? \1, \1 — логирование
- ? \1 — базовый интерфейс состояния
- ? \1, \1 — состояния \1, \1, \1
- ? \1 — интерфейс адаптера рендера
- ? \1, \1 — реализация адаптера на \1
- ? \1, \1 — управление объектом и ввод

7. Реализованная функциональность (результат работы)

В результате выполнения задания получено приложение, которое:

1. Запускает графическое окно.
2. Работает в рамках главного игрового цикла.
3. Вычисляет Δt и выводит его в лог.
4. Поддерживает переключение состояний (Δt , Δt , Δt).
5. Отрисовывает тестовый примитив (квадрат).
6. Позволяет управлять примитивом с клавиатуры и мыши.
7. Выполняет плавные изменения положения/масштаба/поворота с учетом Δt .

8. Листинги наиболее значимых частей кода

В отчет рекомендуется вставить фрагменты следующих файлов:

1. Интерфейс адаптера рендера: Δt
2. Главный цикл и расчет Δt : Δt
3. Базовый интерфейс состояния и реализация состояний: Δt , Δt
4. Обработка ввода с учетом Δt : Δt
5. Реализация рендера через Δt : Δt

9. Скриншоты рабочего приложения

В соответствии с требованиями к отчету необходимо приложить скриншоты.

Рекомендуемый набор:

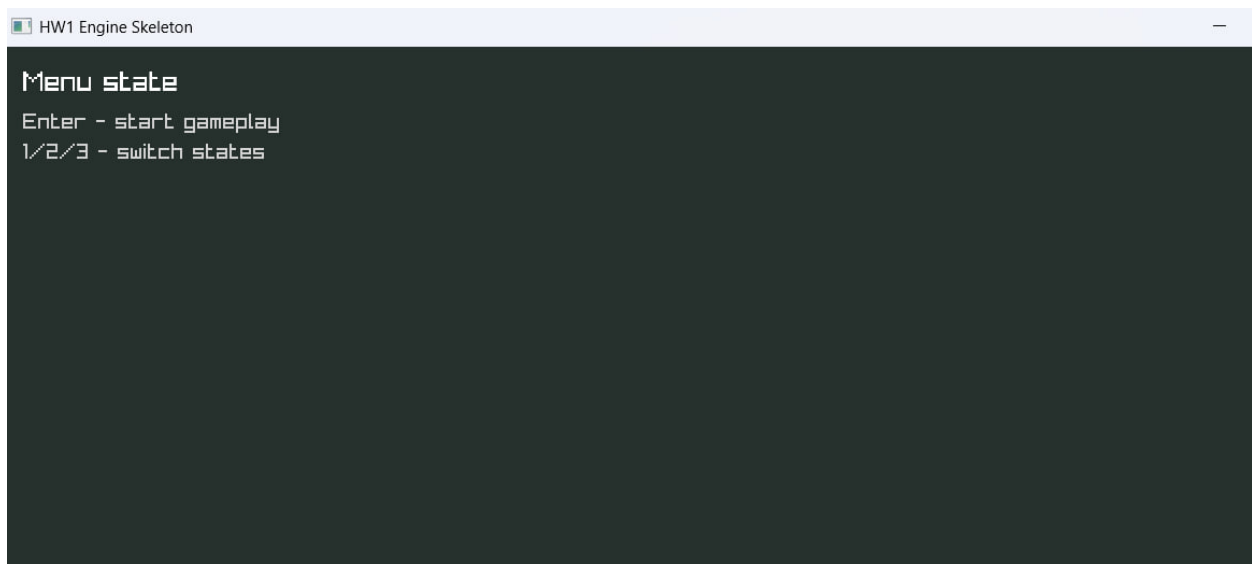
1. Окно приложения в состоянии Δt
2. Окно приложения в состоянии Δt с тестовым примитивом
3. Демонстрация изменения положения/масштаба/поворота
4. Консоль/лог с выводом Δt и переходов между состояниями

Файлы скриншотов в репозитории:

? Δt

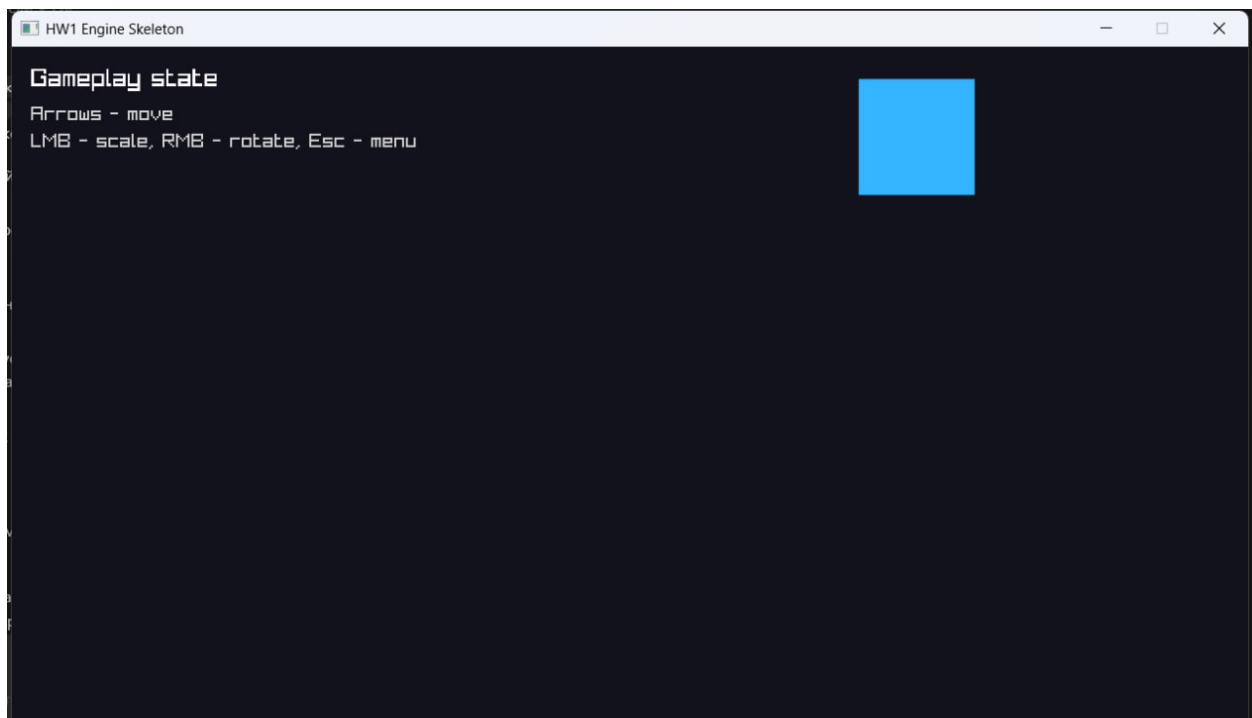
? Δt

Скриншот 1. Состояние меню



Состояние меню

Скриншот 2. Состояние игрового режима с тестовым примитивом



Состояние игрового режима

Дополнительно рекомендуется (при наличии) вставить:

- ? Скриншот 1. Состояние меню.
- ? Скриншот 2. Состояние игрового режима с примитивом.
- ? Скриншот 3. Результат работы обработки ввода.
- ? Скриншот 4. Логирование значений \1.

10. Тестирование и проверка работоспособности

Проверена следующая функциональность:

1. Запуск приложения и создание окна рендера.
2. Переключение между состояниями (\1, \1, \1, \1, \1).
3. Отрисовка тестового примитива в \1.
4. Перемещение объекта стрелками.
5. Масштабирование по ЛКМ.
6. Вращение по ПКМ.
7. Вывод значений \1 в журнал.

Результат: функциональность работает корректно, поведение объекта визуально плавное.

11. Выводы и комментарии по трудностям

В ходе выполнения работы был реализован минимальный каркас игрового движка с возможностью дальнейшего расширения.

Основные результаты:

- ? реализован класс \1 и главный цикл;
- ? добавлен расчет \1 и журналирование;
- ? реализована система состояний;
- ? реализован \1 и конкретный адаптер для \1;
- ? выполнена отрисовка тестового примитива;
- ? реализовано дополнительное задание по обработке ввода клавиатуры и мыши.

Трудности:

- ? при сборке через новую версию CMake возникла несовместимость с \1 (предупреждение/ошибка по старой минимальной версии CMake в зависимом проекте);
- ? проблема была устранена настройкой политики CMake в основном \1.

12. Приложения к отчету

12.1. Исходный код

Репозиторий проекта:

\1

12.2. Конфигурационные файлы

В репозитории присутствуют:

- ? \1
- ? \1
- ? документация в папке \1

12.3. Дополнительные материалы

При необходимости можно приложить:

- ? видео демонстрации работы приложения;
- ? архив со скриншотами;
- ? экспортированный PDF-отчет.

13. Контрольные вопросы (краткие ответы)

1. Чем отличается фиксированный и переменный шаг игрового цикла?

Фиксированный шаг использует постоянный интервал обновления, что удобно для физики и повторяемости.

Переменный шаг использует фактическое Δt между кадрами, что проще для базовой реализации, но требует ограничения слишком больших значений времени.

2. Зачем необходим Δt и можно ли обходиться без него?

Δt уменьшает связанность между игровой логикой и конкретным графическим API.

Без него можно обойтись в маленьком проекте, но код станет хуже расширяться и переноситься на другую библиотеку.

3. Какова роль Δt в управлении физическими процессами и анимациями?

Δt позволяет изменять параметры объектов пропорционально времени, а не количеству кадров, что делает движение более стабильным при разном FPS.

4. Какие преимущества дает использование системы состояний в играх?

Система состояний разделяет поведение приложения по режимам (загрузка, меню, игра), упрощает структуру кода и облегчает добавление новых экранов/режимов.

5. Как технически организовать смену графического API («горячую загрузку»)?

Необходимо работать через абстракцию (Δt) и отделять игровые данные от графических ресурсов.

При смене API создается новый адаптер и выполняется повторная инициализация графических ресурсов, при этом логика игры продолжает работать через общий интерфейс.

Примечание по сдаче

Для финальной версии отчета рекомендуется:

1. Вставить реальные скриншоты в раздел 9.
2. При необходимости перенести текст в шаблон вуза.
3. Экспортировать в PDF и приложить ссылку на репозиторий.