МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИИ МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "САНКТ-ПЕТЕРБУРГСКИЙ

ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М. А. БОНЧ-БРУЕВИЧА"

Факультет инфокоммуникационных сетей и систем

Кафедра сетей связи и передачи данных

ЛАБОРАТОРНАЯ РАБОТА №6

по дисциплине «Процессы жизненного цикла программного обеспечения»

Тестирование программного обеспечения

Выполнил:

студент 4 курса

ИКПИ-14

Хохлов Тихон Владимирович

Санкт-Петербург

2025

# **Введение**

Провести тестирование и анализ проекта по предмету «Процессы жизненного цикла программного обеспечения» студентов группы ИКПИ-11 Крылова А.В., Дунаева В.Е. и студента группы ИКПИ-14 Хохлова Т.В., с точки зрения качества, выявить несоответствия критериям тестируемого проекта и предложить рекомендации по улучшению. Необходимо сформировать по ранее разработанной тестовой документации тест-план, включить в него некоторое подмножество тест-кейсов и выполнить тест-кейсы, представленные в тест-плане.

Критерии к тестируемому проекту взяты из ГОСТ 19.101-77 («Общие требования к программной документации»), ГОСТ 34.602-89 («Техническое задание на создание автоматизированной системы»), ГОСТ 19.104-78 («Основные надписи»), ГОСТ 19.105-78 («Общие требования к построению, изложению и оформлению документов»), ГОСТ 34.603-92 («Испытания и приемка автоматизированных систем»).

**Описание программного проекта**

Техническое задание студенческого проекта «КОМПЬЮТЕРНАЯ ИГРА «БОЯРСКИЙ ТУРНИР»» по дисциплине «Процессы жизненного цикла программного обеспечения» студентов группы ИКПИ-11 Крылова А.В., Дунаева В.Е., ИКПИ-14 Хохлова Т.В.

Проект представляет многопользовательскую игру в жанре RTS. Основная цель приложения — заполнить нишу на рынке мобильных игр после ухода Supercell из России, предоставив альтернативу Clash Royale, используя знакомые элементы геймплея и тематику.

В качестве ключевых функций можно выделить следующие:

1. Сражения (Бой 1 на 1)

2. Сбор эликсира

3. Призыв юнитов

4. Использование специальных способностей

5. Цикличная смена карт

6. Переопределение колод юнитов

Программа разработана на языке Rust в интегрированной среде разработки NeoVIM в операционной системе Arch linux. Также использовались следующие технологии и библиотеки:

* Bevy — это набор библиотек для разработки игр на языке Rust, включает в себя такие модули, как ECS, рендер-граф на основе WebGPU, инджектор зависимостей, сцены, ассет-менеджер, ввод/вывод и прочее

# **Описание используемых**

В качестве тестируемой части приложения выбран этап проведения боя 1 на 1. Скриншот видимого поля с UI для клиента:



*Рисунок 1 — UI двух клиентов*

**Инструменты**

Для реализации автоматизированного тестирования в проекте "Боярский турнир" были использованы следующие инструменты и технологии:

* Встроенный тестовый фреймворк **Rust**

Основой для написания тестов послужил стандартный инструментарий языка **Rust** (**cargo** **test**), обеспечивающий выполнение модульных и интеграционных тестов. Фреймворк включает макросы **assert**!, **assert**\_**eq**! и **assert**\_**ne**! для проверки корректности работы функций и методов.

* **Bevy** **ECS**

Для тестирования игровых систем и их взаимодействия использовался движок **Bevy** и его **Entity** **Component** **System** (**ECS**). Это позволило эмулировать игровые сущности, компоненты и их состояния, не загружая полную игровую среду.

**Подходы**

При написании автотестов использовался метод модульного тестирования и метод белого ящика.

# **Код автотеста**

|  |
| --- |
| **common/src/lib.rs** |
| …  // TESTS  #[cfg(test)]  mod tests {  use super::\*;  // Тест 1: Накопление "эликсира" (Health как аналог)  #[test]  fn test\_mana\_regeneration() {  let mut health = Health::new(100);  // Эмулируем регенерацию маны (увеличение здоровья)  health.0 += 10;  assert\_eq!(health.0, 110);  assert\_eq!(health.1, 100); // Максимум не должен измениться  }  // Тест 2: Использование карт юнитов  #[test]  fn test\_card\_usage() {  let card = Card::Rus;  let placement = ArenaPos(100.0, 200.0);    // Проверяем валидную позицию  assert!(placement.0 < 400.0, "Карта должна размещаться на своей территории");    match card {  Card::Rus => assert!(true),  \_ => panic!("Неожиданный тип карты"),  }  }  // Тест 3: Сражение юнитов  #[test]  fn test\_unit\_combat() {  let attacker = Health::new(100);  let mut target = Health::new(100);    // Эмулируем атаку  target.0 -= 10; // Простой урон    assert\_eq!(target.0, 90);  assert\_eq!(attacker.0, 100); // Здоровье атакующего не изменилось  }  // Тест 4: Сражение башен  #[test]  fn test\_tower\_combat() {  let mut tower = Health::new(500); // Башня  let damage = 30;    tower.0 -= damage;  assert\_eq!(tower.0, 470);  }  // Тест 5: Уничтожение башен  #[test]  fn test\_tower\_destruction() {  let mut tower = Health::new(500);  tower.0 = 0; // Уничтожили башню    assert\_eq!(tower.0, 0);  }  // Тест 6: Завершение боя  #[test]  fn test\_game\_end() {  let king\_tower = Health(0, 1000); // Главная башня уничтожена  assert\_eq!(king\_tower.0, 0, "Игра должна завершиться");  }  // Тест 7: Проверка позиционирования  #[test]  fn test\_position\_validation() {  let valid\_pos = ArenaPos(100.0, 200.0); // Своя территория  let invalid\_pos = ArenaPos(700.0, 200.0); // Вражеская территория    assert!(valid\_pos.0 < 400.0);  assert!(invalid\_pos.0 > 400.0);  }  // Тест 9: Состояния юнитов (UnitState)  #[test]  fn test\_unit\_states() {  let state = UnitState::Attacking;  match state {  UnitState::Attacking => assert!(true),  \_ => panic!("Неверное состояние"),  }  }  // Тест 10: Работа с ArenaPos  #[test]  fn test\_arena\_pos\_math() {  let pos1 = ArenaPos(3.0, 4.0);  let pos2 = ArenaPos(1.0, 2.0);    // Проверяем операции  let diff = pos1 - pos2;  assert\_eq!(diff.0, 2.0);  assert\_eq!(diff.1, 2.0);    // Проверяем расстояние  let distance = pos1.distance(&pos2);  assert!((distance - 2.828427).abs() < 0.0001);  }  } |

# **Отчет о тестировании**

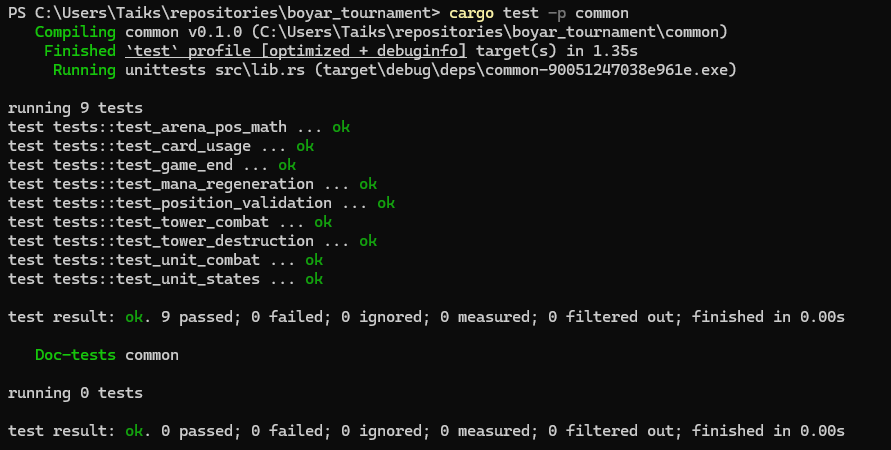
Выполненные тест-кейсы

Список выполненных тест кейсов представлен в таблице 1:

*Таблица 1 – Список используемых тест-кейсов*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ п/п** | **Название тест-кейса** | **Описание** | **Ожидаемый результат** | **Тип теста** |
| 1 | Накопление эликсира | Обновление количества доступного для использования эликсира в определенный момент времени | Кол-во эликсира обновляется со временем | Позитивный |
| 2 | Использование карт юнита | Попытка использовать карту юнита на своей территории | Успешное появление юнита на поле боя | Позитивный |
| 3 | Использование карт заклинания | Попытка использовать карту заклинания на любой территории | Успешное использование заклинания на выбранной территории | Позитивны |
| 4 | Сражение юнитов | При достаточном сближении юнитов должен происходить локальный бой между ними | Успешная реализация локальных боев между юнитами при их сближении друг к другу | Позитивный |
| 5 | Сражение башен | При приближении юнитов к радиусу атаки вражеских башен те должны наносить урон ближайшему юниту до его смерти | Успешная реализация локального боя между башней и приближающихся юнитов | Позитивный |
| 6 | Уничтожение башен | По истечению количества здоровья башен, те удаляются с поля боя с соответствующей анимацией уничтожения | Успешное удаление боевой башни по истечении у той очков здоровья | Позитивный |
| 7 | Проверка уничтожения главной башни (союзной или вражеской) | По истечению количества здоровья у главного союзного/вражеского строения должна высветиться надпись «Бой завершен!» с кнопкой выхода в главное меню | Успешное завершение боя, появление всплывающего окна со сводкой боевой статистики и переходом в главное меню | Позитивный |
| 8 | Проверка постановки юнита на вражескую территорию | Использование карты для постановки юнита на вражеской территории | Отмена появления юнита и возврат карты в руку | Негативный |
| 9 | Проверка использования карты при недостаточном количестве эликсира | Использования карт с определенной стоимостью эликсира при её недостаточном количестве | Отмена использования карты и возврат её в руку | Негативный |

**Результат тестирования**



Данная информация сигнализирует о успешном выполнении всех, использованных нами тестах.

**Выявленные дефекты**

На основании проведенных автотестов дефекты обнаружены не были.

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были дописаны тесты, проверяющие работоспособность систем, связанных с непосредственным проведением боёв 1 на 1.

Этот отчет может быть использован для дальнейшего улучшения приложения и обеспечения его стабильной работы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Отчет по проекту по предмету «Процессы жизненного цикла программного обеспечения» студентов группы ИКПИ-11 Крылова А.В., Дунаева В.Е. и студента группы ИКПИ-14 Хохлова Т.В.
2. Методические материалы для Лабораторной работы №6 по предмету «Процессы жизненного цикла ПО».
3. Тестирование Дизайна Прототипов Как Необходимое Звено Разработки Продукта. <https://cyberleninka.ru/article/n/testirovanie-dizayna-prototipov-kak-neobhodimoe-zveno-razrabotki-produkta> (дата обращения 20.03.2025 г.)
4. Немного о простом. Тест-дизайн. Часть 1. <https://habr.com/ru/articles/462553/> (дата обращения 20.03.2025 г.)