

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

**МОБИЛЬНАЯ 2D-ИГРА ДЛЯ УСТРОЙСТВ
НА ОС ANDROID**

Студент


(подпись, дата)

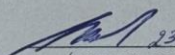
Горбунов Данила Алексеевич
(ФИО полностью)

Руководитель


(подпись, дата)

Кузнецов А.В.
(фамилия, инициалы)

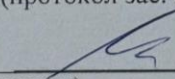
Нормоконтролёр


(подпись, дата)

Бычков А.С.
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026г. № 14)

Заведующий кафедрой
(подпись)


(фамилия, инициалы)

Митрохин М.А.

Работа защищена с отметкой хорошо (протокол заседания ГЭК от

24 июня 2026г. № 47)

Секретарь ГЭК
(подпись)


(фамилия, инициалы)

Гурин Е.И.

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 07 » 04 20 26 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Горбунов Данила Алексеевич гр.22ВВП2 факультета вычислительной техники

2. Тема работы Мобильная 2D-игра для устройств на ОСAndroid

Тема утверждена приказом ПГУ №347/о _____ от "07" _____ апреля _____ 2026 _____ г.

3. Руководитель бакалаврской работы доцент кафедры ВТ ПГУ Кузнецов А.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

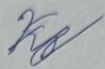
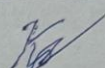
Разработать мобильную 2D-игру в жанре фермы, позволяющую перемещаться по карте, высаживать растения и собирать урожай. Язык программирования GDScript, среда разработки Godot. Программа должна обладать интуитивно понятным интерфейсом и использовать следующие технологии:

1. TileMapLayer
2. AnimatedSprite2D
3. VirtualJoystick
4. YSort
5. CollisionShape2D

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области: изучить существующие аналоги, изучить способы проектирования и разработки мобильной игры. Выбрать среду разработки и язык программирования.

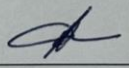
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	46	07.04.2026 – 11.04.2026	
2	Анализ требований	25	12.04.2026 – 14.04.2026	
3	Выбор технологий, применяемых при проектировании и разработке	35	15.04.2026 – 19.04.2026	
4	Создание графического интерфейса	85	20.04.2026 – 01.05.2026	
5	Разработка мобильной игры	93	02.05.2026 – 14.05.2026	
6	Оформление пояснительной записки	40	15.05.2026 – 21.05.2026	

Дата выдачи задания "07" апреля 2026г.

Руководитель бакалаврской работы 
(подпись, дата)

Задание к исполнению принял студент 
(подпись, дата)

Работу к защите допустить
Декан факультета  А.Р. Речнев
24.06.2026 (подпись, дата)

Реферат

Бакалаврская работа содержит 72 листа, 30 рисунков, 0 формул, 1 таблицу, 7 источников.

МОБИЛЬНАЯ ИГРА, 2D, ТЕСТИРОВАНИЕ, TILEMAPLAYER, ANIMATEDSPRITE2D, VIRTUALJOYSTICK, YSORT, COLLISIONSHAPE2D, GODOT, GDSCRIPT, ANDROID

Цель работы – разработать мобильную 2D-игру в жанре RPG-фермы.

Мобильная игра должна позволять пользователям, играя за персонажа, перемещаться по игровой карте, а также реализовать функционал посадки, роста растений и сбора урожая. Помимо этого должна присутствовать система коллизий для корректного перемещения игрока по карте и система сортировки по Y для правильной отрисовки объектов и персонажа поверх друг друга. Помимо этого должен быть рабочий инвентарь и система сохранений.

В конечном итоге была разработана мобильная игра на языке программирования GDScript, которая удовлетворяет всем поставленным задачам. Была создана игровая карта с помощью TileMapLayer, сортировка объектов с помощью YSort, перемещение персонажа с помощью VirtualJoystick и CollisionShape2D, отрисовка анимаций персонажа с помощью AnimatedSprite2D. Само приложение работает на устройствах с ОС Android и разрабатывалось в специальной среде разработки Godot.

					ПГУ 1.090301.34.001 ПЗ			
Изм	Лист	Недокум.	Подпись	Дата	Мобильная 2D-игра для устройств на ОСAndroid Пояснительная записка	Лит.	Лист	Листов
Разраб.		Горбунов Д.А.		23.06.26				
Провер.		Кузнецов А.В.		23.06.26			2	72
Н. Контр.		Бычков А.С.		23.06.26		ФВТ, каф. ВТ гр. 22ВВП2		
Утверд.		Митрохин М.А.		23.06.26				

Содержание

Перечень сокращений и обозначений.....	5
Введение.....	6
1 Мобильная 2D-игра для устройств на ОС Android.....	7
1.1 Анализ темы.....	7
1.2 Анализ аналогов	8
1.2.1 «Hay Day»	8
1.2.2 «FarmVille».....	9
1.2.3 «Stardew Valley».....	10
1.2.4 Вывод.....	11
1.3 Постановка задачи.....	12
2 Выбор инструментов разработки	14
2.1 Выбор среды разработки	14
2.1.1 «Android Studio»	14
2.1.2 «Unity».....	15
2.1.3 «Godot».....	15
2.1.4 Вывод.....	15
2.2 Выбор языка программирования	15
3 Реализация приложения	17
3.1 Описание функционала.....	17
3.2 Описание работы приложения	18
3.3 Архитектура приложения	19
3.3.1 Модуль управления персонажем	19
3.3.2 Модуль инвентаря	20
3.3.3 Модуль пользовательского интерфейса	20
3.3.4 Модуль управления растениями	20
3.3.5 Модуль роста растений.....	22
3.3.6 Модуль обработки взаимодействий.....	22
3.3.7 Модуль отображения графики	24
3.3.8 Модуль анимаций	26

3.3.9	Модуль коллизий	26
3.4	Основная реализация приложения	27
3.4.1	player.tscn.....	27
3.4.2	crops.tscn.....	29
3.4.3	world.tscn	31
4	Тестирование	35
4.1	Передвижение персонажа.....	35
4.2	Ограничение передвижения для неигровых зон	38
4.3	Посадка растений	39
4.4	Сбор урожая	44
4.5	Вывод.....	47
5	Охрана труда.....	48
5.1	Правила и техника безопасности при работе с ПК.....	48
5.2	Правила и техника безопасности при работе со смартфоном	50
	Заключение	52
	Список использованных источников	53
	Приложение А UML-диаграммы.....	54
	Приложение Б Листинг.....	60
	Приложение В Презентация.....	67

Перечень сокращений и обозначений

В данной работе применяются следующие сокращения и обозначения.

ОС – операционная система

ПК – персональный компьютер

ПО – программное обеспечение

RPG – (roleplayinggame) – компьютерная ролевая игра

UI – (от англ. UserInterface) – пользовательский интерфейс

UML – (от англ. UnifiedModelingLanguage) – унифицированный язык моделирования

					ПГУ 1.090301.34.001 ПЗ	Лист
						5
Изм.	Лист	№докум.	Подпись	Дата		

Введение

В настоящее время мобильные устройства являются неотъемлемой частью повседневной жизни человека. Вместе с развитием смартфонов и мобильных технологий активно развивается рынок мобильных приложений, среди которых особое место занимают мобильные игры. Благодаря доступности, удобству использования и широкому распространению устройств на базе операционной системы Android мобильные игры стали одним из самых популярных видов цифровых развлечений.

Разработка мобильных игр представляет собой актуальное направление в сфере информационных технологий, объединяющее программирование, графический дизайн и проектирование пользовательского интерфейса. Особенно востребованными остаются 2D-игры, поскольку они отличаются сравнительно простой реализацией, невысокими требованиями к производительности устройств и удобством использования на мобильных платформах. Кроме того, двухмерные игры позволяют сосредоточиться на игровом процессе и удобстве взаимодействия пользователя с приложением.

Цели выпускной квалификационной работы:

- исследовать тему использованию данного продукта;
- определить функциональность и интерфейс мобильного приложения;
- разработать современную мобильную 2D-игру;
- провести тестирование ПО;
- предложить рекомендации по развитию и расширению игровых механик для дальнейших обновлений приложения;
- проанализировать полученные результаты и сделать выводы о потенциале разработки данного приложения.

1 Мобильная 2D-игра для устройств на ОС Android

1.1 Анализ темы

Тема разработки мобильной 2D-игры для устройств на базе операционной системы Android является одной из наиболее актуальных в современной сфере информационных технологий и цифровых развлечений. Это обусловлено стремительным развитием мобильной индустрии, а также ростом популярности мобильных приложений, в частности игр, среди пользователей по всему миру.

В последние годы мобильный гейминг занял лидирующие позиции в игровой индустрии, опередив по объёму аудитории и доходам традиционные платформы, такие как персональные компьютеры и игровые консоли. Значительная часть этого успеха связана с широким распространением устройств на базе Android, который является самой массовой мобильной операционной системой. Благодаря своей открытости и доступности платформа предоставляет разработчикам широкие возможности для создания и распространения собственных продуктов через такие сервисы, как Google Play.

Особое место в мобильной разработке занимают 2D-игры. Несмотря на активное развитие трёхмерной графики, двумерные игры продолжают пользоваться устойчивым спросом. Это объясняется их относительной простотой в разработке, меньшими требованиями к аппаратным ресурсам устройств и возможностью быстрого создания прототипов. 2D-графика позволяет сосредоточиться на игровом процессе, механиках и пользовательском опыте, что особенно важно для начинающих разработчиков и небольших команд.

Кроме того, 2D-игры обладают высокой универсальностью. Они способны корректно работать на широком спектре устройств, включая бюджетные смартфоны, что существенно расширяет потенциальную аудиторию. Успех таких проектов, как AngryBirds или GeometryDash, демонстрирует, что даже относительно простые по технической реализации игры могут достигать

огромной популярности при наличии продуманного игрового процесса и привлекательного дизайна.

С технологической точки зрения разработка 2D-игр для Android также является удобной и доступной. Современные игровые движки, такие как Unity и Godot, предоставляют широкий набор инструментов для создания игр различной сложности. Они позволяют значительно ускорить процесс разработки, упростить работу с графикой, физикой и пользовательским интерфейсом, а также обеспечивают возможность кроссплатформенного распространения.

Основными пользователями такой игры являются обычные владельцы смартфонов — люди разных возрастных категорий, которые используют мобильные устройства для досуга. Для них игра выступает средством развлечения, позволяющим скоротать время в дороге, в очередях или во время отдыха. Простота управления и доступность 2D-игр делают их особенно привлекательными для широкой аудитории, включая детей, подростков и взрослых, не имеющих большого игрового опыта.

1.2 Анализ аналогов

При разработке мобильной игры в жанре «ферма» важно учитывать уже существующие проекты, так как этот жанр сформировался давно и имеет устойчивые механики. На рынке существует несколько ярких представителей, такие как:

- Hay Day;
- FarmVille;
- Stardew Valley.

1.2.1 «Hay Day»

Приложение представлено на рисунке 1.

Одним из самых известных представителей данного жанра является Hay Day. Это классическая мобильная ферма, в которой игрок управляет хозяйством: выращивает культуры, разводит животных, производит товары и торгует с

					ПГУ 1.090301.34.001 ПЗ	Лист
						8
Изм.	Лист	№докум.	Подпись	Дата		

другими игроками. Игра отличается простой и понятной механикой, постепенным развитием и акцентом на спокойный, расслабляющий игровой процесс. Важной особенностью является социальное взаимодействие — возможность помогать другим игрокам, обмениваться ресурсами и участвовать в совместных событиях.

Главными преимуществами этой игры являются приятный визуальный стиль, интуитивно понятный геймплей и высокая степень вовлечённости пользователей. Игра не требует сложного обучения и подходит для широкой аудитории. Однако у неё есть и недостатки: со временем игровой процесс становится однообразным, а развитие замедляется без вложения реальных средств. Также некоторые игроки отмечают зависимость от таймеров и необходимость регулярно заходить в игру.

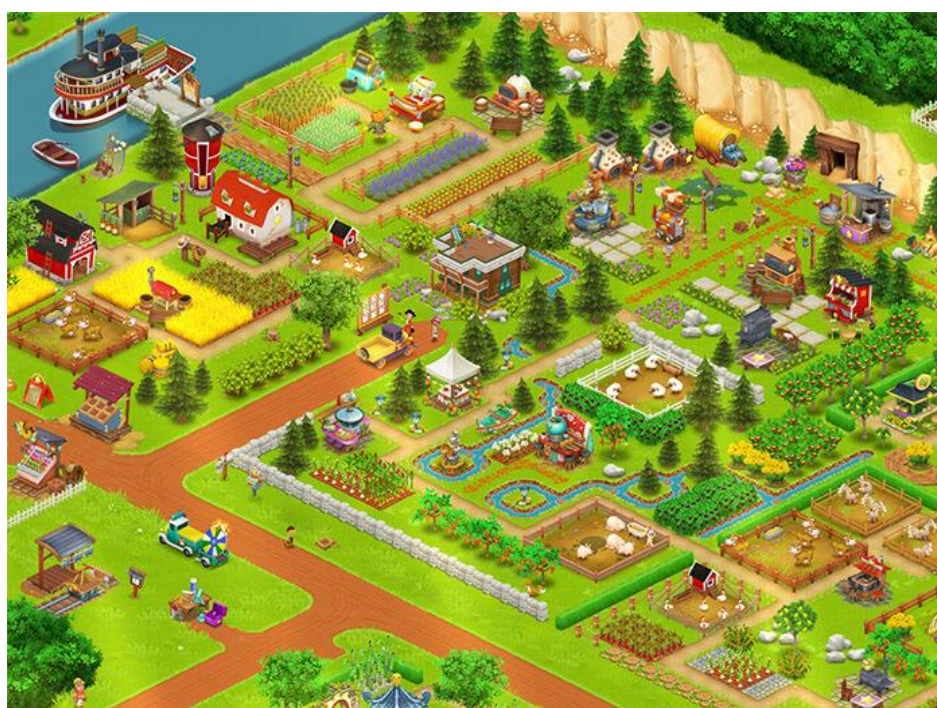


Рисунок 1 – Игра «Hay Day»

1.2.2 «FarmVille»

Приложение представлено на рисунке 2.

FarmVille – одна из первых игр, популяризовавших жанр фермы. Основной геймплей включает посадку и сбор урожая, развитие хозяйства и

выполнение заданий. Игра активно использует социальные элементы и взаимодействие с другими игроками.

Преимуществом FarmVille является разнообразие контента и наличие заданий, которые направляют игрока и создают ощущение прогресса. Также игра предлагает множество событий и активностей. Однако её главным недостатком считается агрессивная монетизация: многие элементы игры ускоряются или становятся доступными только за реальные деньги. Кроме того, часть пользователей отмечает, что игра может быстро становиться скучной и требующей однотипных действий.



Рисунок 2 – Игра «FarmVille»

1.2.3 «Stardew Valley»

Приложение представлено на рисунке 3.

В отличие от классических «ферм», Stardew Valley предлагает более глубокий игровой процесс: помимо сельского хозяйства, игрок может заниматься рыбалкой, добычей ресурсов, исследованием мира и взаимодействием с персонажами. Игра имеет сюжетные элементы и систему развития отношений с NPC.

Главным преимуществом Stardew Valley является глубина геймплея и высокая степень свободы действий. Игрок не ограничен только механикой выращивания растений, что делает игровой процесс разнообразным и менее однообразным. Также важным плюсом является отсутствие навязчивой монетизации — игра приобретается один раз и не требует дополнительных вложений. Однако к недостаткам можно отнести более высокий порог входа и сравнительно сложное управление для мобильных устройств, особенно для новичков.



Рисунок 3 – Игра «Stardew Valley»

1.2.4 Вывод

Результаты анализа аналогов можно увидеть на таблице 1.

Проанализировав данные игры можно сделать вывод, что у большинства игр, ориентированных на большую аудиторию низкий порог входа. среднее разнообразие, и высокая зависимость от таймеров, которая искусственно повышает вовлеченность игрока, заставляя часто заходить в игру, чтобы не замедлить прогресс. Учитывая это, создание игры, в которой будет низкий порог входа, при этом добавлено больше разнообразия и исключены таймеры, а

прогресс ограничивается лишь временем, потраченным пользователем на игру, можно считать актуальным.

Таблица 1 – Достоинства и недостатки игр

Игра	Hay Day	FarmVille	Stardew Valley
Простота освоения	Просто	Просто	Средне
Свобода действий	Ограниченная	Ограниченная	Высокая
Разнообразие	Среднее	Среднее	Очень высокое
Монетизация	Средняя	Высокая	Отсутствует
Зависимость от таймеров	Высокая	Высокая	Средняя

1.3 Постановка задачи

Необходимо разработать мобильную 2D-игру для телефонов на базе операционной системы Android, представляющее собой интерактивную игровую среду, в которой пользователь управляет виртуальной фермой, выращивает сельскохозяйственные культуры, разводит животных и развивает своё хозяйство. Среди мобильных устройств операционная система Android является лидером по их количеству и занимает 70% от глобального рынка. По этой причине для разработки была выбрана именно эта ОС.

В приложении должно быть реализовано:

- интуитивно понятный интерфейс;
- управление персонажем;
- пиксельная 2D-графика;
- системы добычи разных ресурсов;
- система прогрессии с разблокировкой нового контента;
- система автоматических сохранений прогресса.

Минимальные системные требования:

- операционная система Android 8.0;
- 1 ГБ ОЗУ.

Основная цель проекта – обеспечить удобный и увлекательный игровой процесс, дать возможность расслабиться и отдохнуть, предоставить бесплатную альтернативу платным играм.

					ПГУ 1.090301.34.001 ПЗ	Лист
						13
Изм.	Лист	№докум.	Подпись	Дата		

2 Выбор инструментов разработки

Для разработки качественного продукта и оптимизации процесса разработки необходимо выбрать правильные инструменты. Это помогает избежать проблем при производстве и использовании продукта, а также позволяет сократить время и средства на его создание.

2.1 Выбор среды разработки

Android является одной из самых популярных операционных систем для мобильных устройств. Поэтому необходимо выбрать подходящую среду, которая будет наиболее эффективна для разработки приложения, которое, в свою очередь, сможет корректно работать на большинстве современных устройств.

Среды разработки, которые позволяют создать мобильные игры для устройств на ОС Android представлены ниже:

- AndroidStudio;
- Unity;
- Godot.

Для того, чтобы решить, какая среда больше всего подходит для разработки приложения, проведем анализ и выявим особенности каждой из них.

2.1.1 «Android Studio»

Android Studio — это официальная среда разработки для создания приложений под Android. Она предназначена в первую очередь для разработки мобильных приложений на Java и Kotlin. Android Studio предоставляет инструменты для создания интерфейсов, работы с базами данных и тестирования приложений. Однако для разработки игр, особенно 2D-игр, данная среда менее удобна, поскольку не содержит встроенных игровых инструментов: разработчику приходится самостоятельно реализовывать анимации, игровую физику и систему рендеринга. Поэтому Android Studio больше подходит для обычных приложений, чем для игровых проектов.

2.1.2 «Unity»

Unity — это популярный игровой движок, широко используемый для создания 2D- и 3D-игр. Он обладает большим количеством встроенных инструментов, поддерживает экспорт на Android и имеет обширное сообщество разработчиков. Unity подходит для создания мобильных игр, однако имеет высокий порог входа и имеет множество платных функций. Движок требует больше системных ресурсов и ориентирован в том числе на сложные 3D-проекты, что делает его менее удобным для простой мобильной 2D-игры.

2.1.3 «Godot»

Godot — это современный игровой движок с открытым исходным кодом, предназначенный для разработки 2D- и 3D-игр. Он отличается лёгкостью, простотой освоения и удобной архитектурой. Godot имеет встроенные инструменты для работы с 2D-графикой, анимацией, интерфейсом и физикой, благодаря чему хорошо подходит для разработки мобильных игр. Дополнительным преимуществом является поддержка экспорта на Android и отсутствие платных лицензий.

2.1.4 Вывод

Проанализировав все варианты было принято решение выбрать Godot в качестве среды разработки, потому что, благодаря его функциональности и простоте, получится написать безопасное, компактное и не затратное приложение.

2.2 Выбор языка программирования

После выбора среды разработки важно правильно выбрать подходящий язык программирования. GodotEngine поддерживает несколько языков программирования, представленных ниже:

- GDScript;
- C#;
- C++.

Рассмотрим GDScript - встроенный язык Godot, созданный специально для работы с этим движком. По синтаксису он напоминает Python, благодаря чему отличается простотой и удобством. GDScript тесно интегрирован в движок и хорошо подходит для быстрой разработки игровой логики. C# и C++ не являются встроенными языками и интегрируются в систему при помощи платформы .NET и системы GDExtension соответственно. Они предназначены для расширения возможностей игрового движка и необходимы для более объёмных проектов.

По итогам анализа поддерживаемых языков программирования был выбран GDScript, потому что он был специально разработан для этой среды и идеально подходит для разработки данного приложения.

					ПГУ 1.090301.34.001 ПЗ	Лист
						16
Изм.	Лист	№докум.	Подпись	Дата		

3 Реализация приложения

Перед началом разработки ПО следует разобраться в архитектуре и функционале разрабатываемого приложения. Так нам потребуется рассмотреть такие важные аспекты разработки приложения как описание работы приложения, функционал и общую структуру приложения.

3.1 Описание функционала

Разрабатываемое приложение представляет собой двухмерную мобильную игру в жанре фермерского симулятора с видом сверху. Основной игровой процесс заключается в выращивании сельскохозяйственных культур, управлении ресурсами и взаимодействии с игровым миром.

К основным функциям приложения относятся:

- передвижение персонажа по игровой карте;
- отображение анимации движения и состояния покоя персонажа;
- посадка различных видов сельскохозяйственных услуг;
- автоматическое изменение внешнего вида растений по мере их роста;
- сбор созревшего урожая;
- хранение игровых предметов в инвентаре;
- выбор активного предмета при помощи панели быстрого доступа;
- отображение количества ресурсов в каждой ячейке инвентаря;
- поддержка сенсорного управления посредством виртуального джойстика.

Варианты использования данного приложения представлены на диаграмме на рисунке 4.

Диаграмма вариантов использования



Рисунок 4 – UML-диаграмма вариантов использования для пользователей

3.2 Описание работы приложения

После запуска приложения пользователю отображается игровая сцена, содержащая карту местности и персонажа. Игрок получает возможность перемещать персонажа по территории игрового мира. Для управления используется либо клавиатура виртуальный джойстик, предназначенный для мобильных устройств.

В процессе игры пользователь может подходить к участкам земли, предназначенным для выращивания растений. Такие участки заранее отмечены специальными тайлами и могут использоваться для посадки культур.

При выборе соответствующего предмета в инвентаре и нажатии на свободную грядку создаётся новый экземпляр растения. Каждое растение содержит собственные параметры роста, информацию о стадиях развития и типе получаемого урожая.

После посадки начинается процесс роста растения. Рост осуществляется автоматически и состоит из нескольких стадий. Каждая стадия характеризуется

отдельным графическим изображением. По истечении заданного времени растение переходит к следующей фазе развития, а после достижения последней стадии становится доступным для сбора.

После взаимодействия с полностью выросшим растением оно удаляется с игрового поля, а соответствующий ресурс добавляется в инвентарь игрока. Таким образом реализуется циклический игровой процесс, состоящий из посадки, ожидания созревания и сбора урожая.

3.3 Архитектура приложения

Приложение построено по модульному принципу. Отдельные подсистемы отвечают за различные аспекты игрового процесса. В состав программы входят следующие модули:

- модуль управления персонажем;
- модуль инвентаря;
- модуль пользовательского интерфейса;
- модуль управления растениями;
- модуль роста растений;
- модуль обработки взаимодействий;
- модуль отображения графики;
- модуль анимаций;
- модуль коллизий.

3.3.1 Модуль управления персонажем

Персонаж является одним из основных объектов приложения. Для него реализована система движения по четырём направлениям: вверх, вниз, влево и вправо. В зависимости от направления движения воспроизводится соответствующая анимация.

При отсутствии ввода персонаж переходит в состояние покоя. Для каждого направления предусмотрены отдельные анимации ожидания, благодаря чему обеспечивается более естественное отображение игрового процесса.

Передвижение реализовано с использованием физической системы движка Godot, что позволяет корректно обрабатывать столкновения и взаимодействия с окружающей средой.

3.3.2 Модуль инвентаря

Для хранения игровых ресурсов реализован инвентарь, расположенный в нижней части экрана. Инвентарь состоит из пяти ячеек, каждая из которых отображает определённый предмет. Внутри ячеек располагаются графическое изображение предмета и числовой индикатор количества экземпляров данного ресурса.

Пользователь может выбрать активный предмет посредством нажатия на соответствующую ячейку. Выбранная ячейка визуально выделяется, что позволяет определить текущий активный предмет.

При изменении количества ресурсов информация в интерфейсе обновляется автоматически.

3.3.3 Модуль пользовательского интерфейса

Интерфейс приложения построен на основе элементов управления движка Godot и располагается поверх игрового мира. Благодаря использованию отдельного слоя интерфейс остаётся неподвижным независимо от положения камеры.

Основными элементами интерфейса являются:

- панель быстрого доступа;
- кнопки выбора предметов;
- счётчики количества ресурсов;
- виртуальный джойстик управления.

Интерфейс адаптирован для сенсорных экранов и обеспечивает удобное взаимодействие с приложением на мобильных устройствах.

3.3.4 Модуль управления растениями

В приложении реализована универсальная система выращивания культур. Каждое растение описывается отдельным набором параметров, включающим:

					ПГУ 1.090301.34.001 ПЗ	Лист
						20
Изм.	Лист	№докум.	Подпись	Дата		

- название культуры;
- количество стадий роста;
- продолжительность каждой стадии;
- графические координаты изображений в атласе текстур;
- тип получаемого урожая.

Посадка осуществляется только при соблюдении нескольких условий:

- игрок должен находиться на допустимом расстоянии от грядки;
- выбранная клетка должна относиться к категории пригодных для посадки;
- на клетке не должно находиться другое растение;
- в инвентаре должно присутствовать необходимое количество семян.

При выполнении всех условий создаётся объект растения, а количество соответствующего ресурса в инвентаре уменьшается. Диаграмма системы посадки растений представлена на рисунке 5.

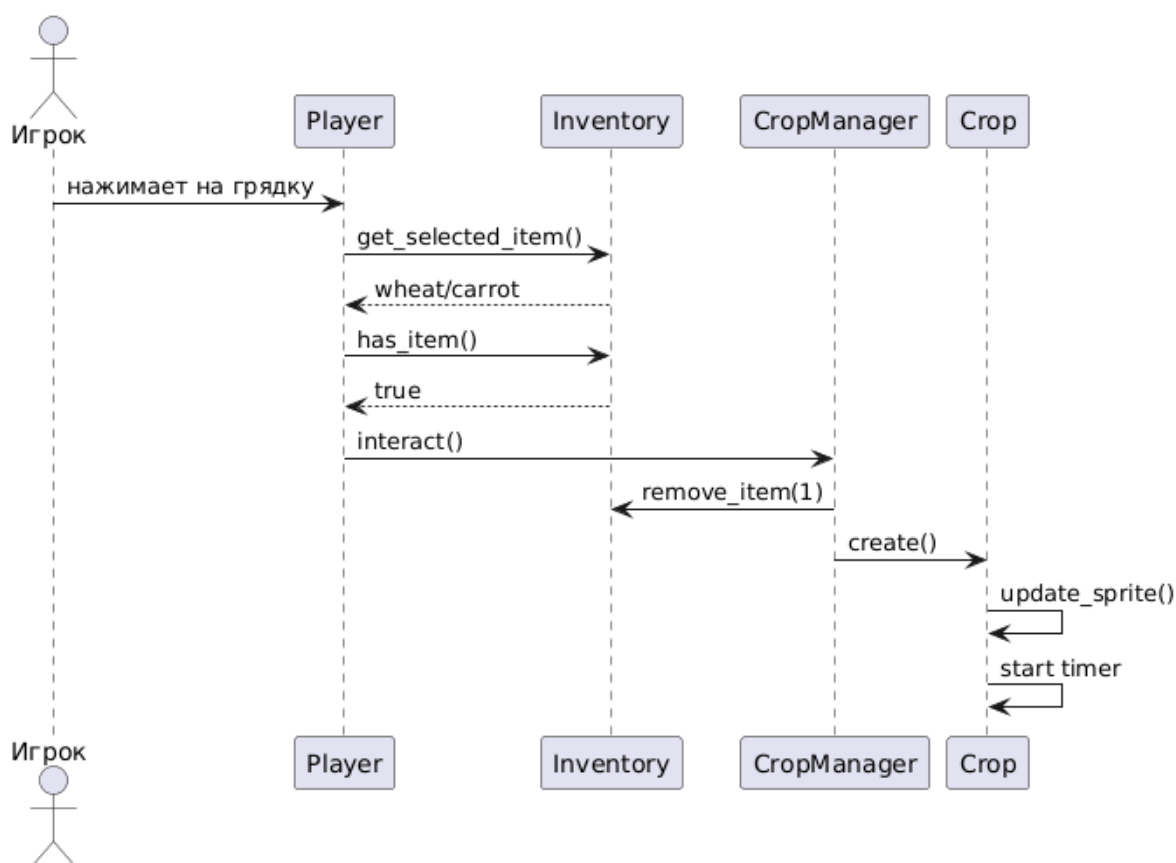


Рисунок 5 – UML-диаграмма системы посадки растений

При взаимодействии с созревшим растением происходит его удаление из игрового мира. После этого игрок получает соответствующий урожай, количество которого автоматически увеличивается в инвентаре.

Данная система обеспечивает непрерывность игрового процесса и позволяет организовать цикл производства ресурсов.

3.3.5 Модуль роста растений

Каждое растение содержит собственный таймер роста. По истечении определённого промежутка времени происходит переход к следующей стадии развития.

Смена стадий сопровождается изменением внешнего вида растения. Для отображения используются различные области атласа текстур. Благодаря этому обеспечивается визуальное отображение процесса роста без необходимости создания отдельных файлов изображений для каждой стадии.

После достижения последнего этапа растение считается полностью созревшим и может быть собрано игроком. Диаграмму состояний растений можно увидеть на рисунке 6.

3.3.6 Модуль обработки взаимодействий

Модуль обработки взаимодействий предназначен для обеспечения связи между игроком и объектами игрового мира. Основной задачей данного модуля является обработка действий пользователя и выполнение соответствующих игровых операций.

Работа модуля начинается с получения информации о действиях игрока. В зависимости от используемого устройства это могут быть нажатия клавиш, касания экрана или сигналы, поступающие от виртуального джойстика. Полученные данные анализируются и преобразуются в игровые действия.

Одной из функций модуля является обработка взаимодействия персонажа с игровыми объектами. Перед выполнением действия производится проверка ряда условий, включая расстояние между персонажем и объектом, доступность объекта для взаимодействия и текущее состояние игрового мира.

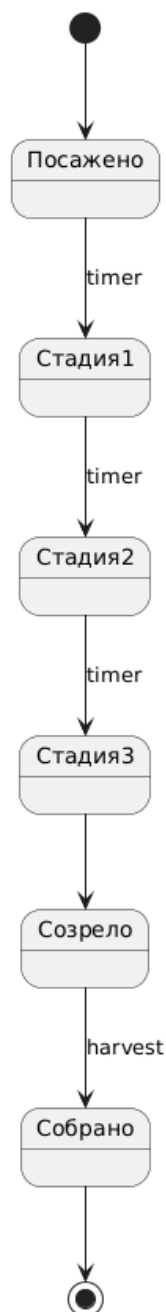


Рисунок 6 – UML-диаграмма состояний растений

При попытке посадки растения модуль определяет координаты клетки, по которой было выполнено нажатие, и проверяет возможность выполнения операции. Если клетка относится к категории пригодных для посадки и на ней отсутствуют другие растения, создаётся новый экземпляр культуры. В противном случае действие игнорируется.

Во время сбора урожая модуль определяет, достигло ли растение последней стадии роста. Если растение полностью созрело, происходит удаление объекта растения из игрового мира и добавление соответствующих ресурсов в инвентарь игрока. Система сбора урожая представлена на диаграмме на рисунке 7.

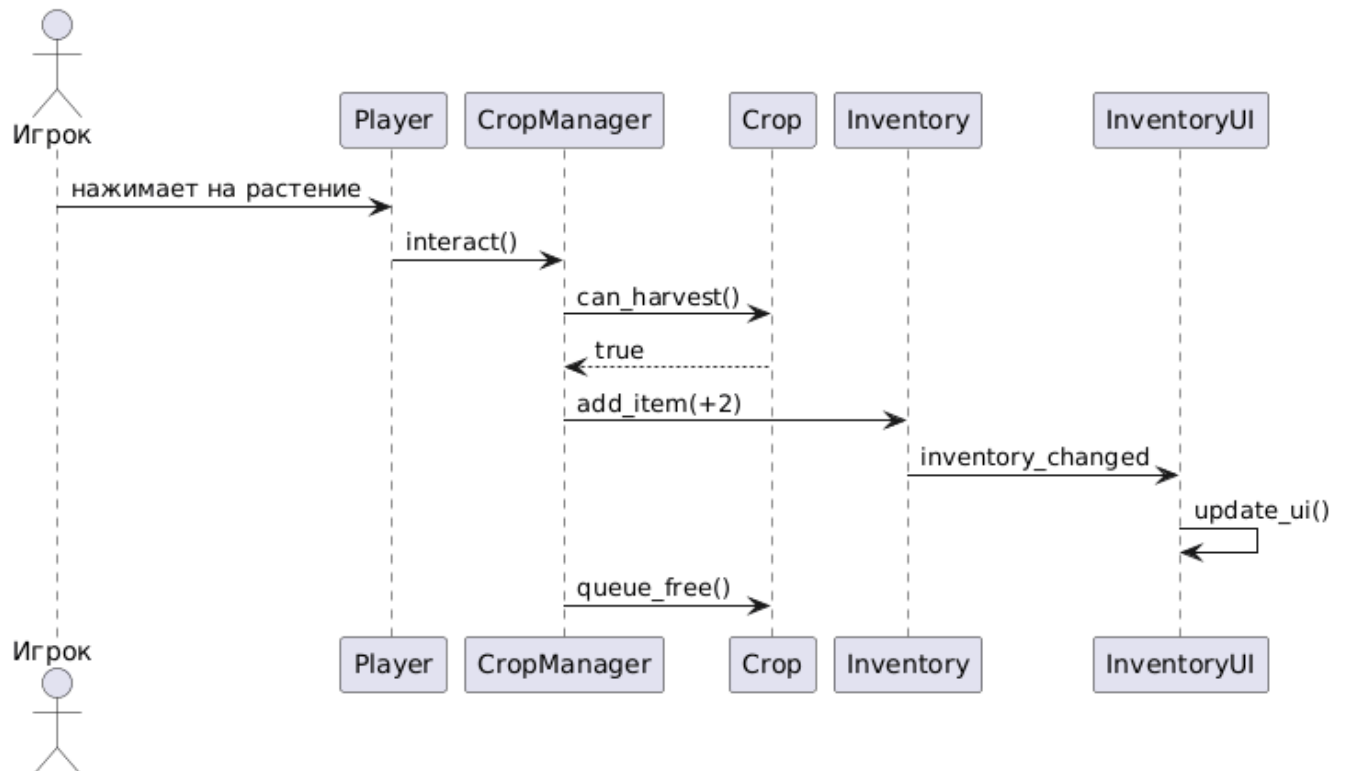


Рисунок 7 – UML-диаграмма системы сбора урожая

Кроме того, модуль обеспечивает взаимодействие с пользовательским интерфейсом. Нажатие на ячейки инвентаря приводит к изменению активного предмета, который впоследствии используется при выполнении игровых действий.

Благодаря централизованной обработке взаимодействий обеспечивается согласованная работа всех игровых подсистем и поддерживается целостность игрового процесса.

3.3.7 Модуль отображения графики

Модуль отображения графики отвечает за визуальное представление объектов игрового мира и пользовательского интерфейса.

Основу графической системы составляют тайловые карты, используемые для построения окружающей среды. С их помощью формируются элементы ландшафта, участки земли, дорожки и другие объекты игрового пространства. Использование тайлового подхода позволяет значительно снизить объём используемой памяти и повысить производительность приложения.

Для хранения изображений применяется технология атласов текстур. Несколько графических элементов объединяются в одно большое изображение, после чего отдельные фрагменты атласа используются для отображения конкретных объектов. Такой подход позволяет уменьшить количество операций загрузки графических ресурсов и повысить эффективность рендеринга.

Отображение растений осуществляется с использованием отдельных сцен, содержащих спрайты и вспомогательные компоненты. В зависимости от стадии роста автоматически изменяется область атласа текстур, используемая для отображения растения.

Для правильного расположения объектов относительно друг друга применяется сортировка по координате Y. Данный механизм позволяет создавать эффект глубины изображения, при котором объекты, расположенные ниже на экране, отображаются поверх объектов, находящихся выше. Благодаря этому обеспечивается естественное перекрытие персонажа и элементов окружающей среды.

Помимо игровых объектов, модуль отвечает за визуализацию пользовательского интерфейса. Панель инвентаря, счётчики ресурсов, кнопки управления и другие элементы располагаются на отдельном слое, независимом от игровой камеры. Это позволяет сохранять постоянное положение интерфейса на экране независимо от перемещения персонажа.

Также модуль обеспечивает отображение текстовой информации, значков предметов и числовых показателей количества ресурсов.

3.3.8 Модуль анимаций

Модуль анимации предназначен для повышения наглядности игрового процесса и обеспечения визуальной обратной связи между приложением и пользователем.

Одной из основных функций модуля является управление анимациями персонажа. Для каждого направления движения предусмотрен отдельный набор кадров, что позволяет отображать перемещение персонажа вверх, вниз, влево и вправо. Кроме того, реализованы анимации состояния покоя, соответствующие последнему направлению движения.

Во время перемещения происходит непрерывное воспроизведение кадров анимации, создающее иллюзию движения. После прекращения перемещения воспроизведение соответствующей анимации завершается, а персонаж переходит в состояние ожидания.

Отдельным направлением работы модуля является анимация роста растений. В процессе роста культура проходит несколько стадий развития, каждая из которых характеризуется собственным графическим представлением. Переход между стадиями осуществляется автоматически через определённые промежутки времени.

Использование нескольких последовательных изображений позволяет визуально отображать процесс развития растений и делает игровой процесс более наглядным и реалистичным.

Модуль анимации тесно взаимодействует с системой отображения графики и системой управления игровыми объектами. Изменение состояния объекта автоматически приводит к изменению его визуального представления.

3.3.9 Модуль коллизий

Модуль коллизий предназначен для обработки физических взаимодействий между объектами игрового мира и предотвращения некорректного перемещения персонажа. Основной задачей данного модуля является контроль допустимых

перемещений игровых объектов и обеспечение реалистичного поведения персонажа в окружающей среде.

Каждый объект окружающего мира, взаимодействующий с персонажем, обладает собственной областью столкновения. Такие области формируются при помощи специальных коллизионных фигур, описывающих границы объектов. В качестве препятствий выступают элементы ландшафта, строения, декоративные объекты и другие элементы игровой карты.

Во время движения персонажа модуль коллизий непрерывно анализирует наличие пересечений между его областью столкновения и коллизионными областями окружающих объектов. В случае обнаружения пересечения дальнейшее перемещение в соответствующем направлении ограничивается. Благодаря этому предотвращается прохождение персонажа сквозь стены, деревья и другие препятствия.

3.4 Основная реализация приложения

Для обеспечения корректной структуры приложения следует поделить его на несколько частей и рассмотреть их по отдельности.

Приложение было поделено на такие сцены как:

- player.tscn;
- crops.tscn;
- world.tscn.

3.4.1 player.tscn

Сцена Player предназначена для реализации игрового персонажа и содержит всю логику его перемещения и взаимодействия с окружающим миром.

Корневым узлом сцены является объект CharacterBody2D, который обеспечивает поддержку встроенной физической системы движка Godot и позволяет использовать механизм обработки столкновений.

В состав сцены входят следующие узлы:

- AnimatedSprite2D, отвечающий за отображение анимации персонажа;

– CollisionShape2D, определяющий область столкновения.

Структуру этой сцены можно увидеть на диаграмме на рисунке 8.

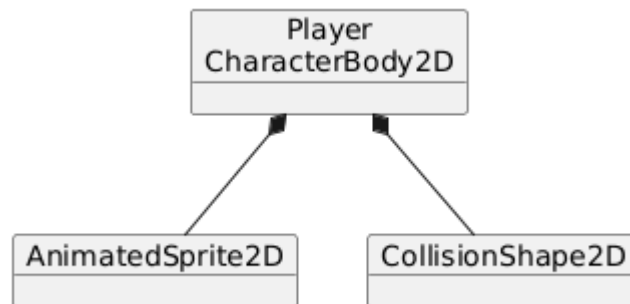


Рисунок 8 – Диаграмма структуры сцены player

К корневому узлу привязан скрипт player.gd, в котором реализовано управление персонажем. В процессе выполнения функции `_physics_process()` осуществляется проверка состояния клавиш управления и формирование вектора скорости движения. После вычисления скорости вызывается функция `move_and_slide()`, выполняющая перемещение объекта с учётом коллизий.

Листинг 1. Функция `_physics_process()`:

```
func _physics_process(delta: float) -> void:
    if Input.is_action_pressed("up"):
        move_up()
    elif Input.is_action_pressed("down"):
        move_down()
    elif Input.is_action_pressed("left"):
        move_left()
    elif Input.is_action_pressed("right"):
        move_right()
    else:
        idle()

    move_and_slide()
```

Дополнительно сцена содержит обработчик пользовательского ввода `_input()`, который отвечает за взаимодействие с грядками. При нажатии на игровое поле вычисляются мировые координаты курсора, преобразуются в координаты тайловой карты и выполняется проверка расстояния между персонажем и выбранной клеткой.

При выполнении всех условий управление передаётся модулю CropManager, который создаёт новый экземпляр растения либо выполняет сбор урожая.

Листинг 2. Функция _input().

```
func _input(event):  
  
    if event.is_action_pressed("left_click"):  
  
        var world_position = get_global_mouse_position()  
        var tile_position = farm_layer.local_to_map(  
            farm_layer.to_local(world_position)  
        )  
  
        if can_reach_tile(tile_position):  
  
            match inventory.get_selected_item():  
  
                "wheat":  
                    if inventory.has_item("wheat"):  
  
crop_manager.interact(tile_position, wheat_data)  
  
                "carrot":  
                    if inventory.has_item("carrot"):  
  
crop_manager.interact(tile_position, carrot_data)
```

3.4.2 crops.tscn

Сцена Crop предназначена для реализации механизма роста растений и представляет собой отдельный объект, создаваемый динамически во время выполнения программы.

Корневым элементом сцены является узел Node2D. Данный узел определяет положение растения в игровом мире и участвует в механизме сортировки объектов по координате Y.

Внутри сцены располагается узел Sprite2D, предназначенный для отображения графического представления растения. Для хранения изображений используется атлас текстур, содержащий все стадии роста культур.

Также в состав сцены входит узел Timer, обеспечивающий работу механизма роста растения. После создания экземпляра сцены таймер

автоматически запускается и через заданные промежутки времени вызывает обработчик сигнала timeout.

Структуру этой сцены можно увидеть на диаграмме на рисунке 9.

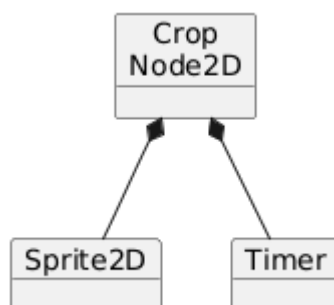


Рисунок 9 – Диаграмма структуры сцены crops

Логика работы растения реализована в скрипте Crop.gd. В момент создания объекта выполняется инициализация параметров растения и отображение начальной стадии развития.

После срабатывания таймера происходит увеличение номера текущей стадии роста. Далее вызывается функция `update_sprite()`, которая изменяет отображаемую область атласа текстур и тем самым обновляет внешний вид растения.

Листинг 3. Функция `update_sprite()`:

```
func update_sprite():

    var coord = crop_data.stage_coords[stage]

    var texture = AtlasTexture.new()

    texture.atlas = atlas_texture

    texture.region = Rect2(
        coord.x * 16,
        coord.y * 16,
        16,
        16
    )

    sprite.texture = texture
```

Переход между стадиями выполняется до тех пор, пока не будет достигнута последняя стадия роста. После этого таймер останавливается, а растение переходит в состояние готовности к сбору урожая.

Для определения возможности сбора используется функция `can_harvest()`, возвращающая логическое значение. Если растение достигло последней стадии развития, функция возвращает значение `true`, что позволяет модулю `CropManager` выполнить сбор урожая.

Сцена `Crop` является универсальной и используется для всех видов растений. Различия между культурами определяются не структурой сцены, а параметрами объекта `CropData`, который передаётся экземпляру растения при его создании. Благодаря такому подходу одна и та же сцена может использоваться для выращивания различных культур, отличающихся временем роста, количеством стадий развития и графическим представлением.

3.4.3 world.tscn

Сцена `World` является основной сценой приложения и представляет собой центральный элемент, объединяющий все игровые подсистемы. Именно данная сцена загружается при запуске приложения и обеспечивает совместную работу объектов игрового мира.

В качестве корневого элемента используется узел `Node2D`, к которому подключаются все остальные компоненты игры.

В состав сцены входят тайловые карты, предназначенные для формирования окружающего мира. Использование нескольких слоёв `TileMapLayer` позволяет разделить различные категории объектов. Первый слой используется для отображения ландшафта и грядок, второй – для отображения декораций и растений, размещаемых в процессе игры, третий – для отображения домика и палаток, четвертый – для отображения деревьев.

Внутри сцены располагается экземпляр сцены `Player`, отвечающий за управление персонажем. Положение персонажа изменяется независимо от остальных объектов, а взаимодействие с игровым миром осуществляется

посредством обращения к другим узлам сцены. К этому экземпляру сцены привязан узел Camera2D, устанавливающий положение камеры на карте и привязывающий камеру к игроку.

Также в составе сцены присутствует узел CropManager, в скрипте которого выполняются функции управления всеми растениями, находящимися на игровом поле. Данный узел осуществляет создание новых экземпляров растений, хранение информации о посаженных культурах и обработку сбора урожая функцией interact().

Листинг 4. Функция interact().

```
func interact(tile_position: Vector2i, crop_data: CropData):
    if tile_position in crops:

        var crop = crops[tile_position]

        if crop.can_harvest():
            inventory.add_item( crop.crop_data.harvest_item,
2 )

            print("Получен предмет:",
crop.crop_data.harvest_item)

            crops.erase(tile_position)

            crop.queue_free()

        return
```

Для хранения ресурсов игрока используется узел Inventory. В его скрипте реализованы структуры данных, отвечающие за количество предметов, а также методы добавления и удаления ресурсов.

Пользовательский интерфейс располагается внутри узла CanvasLayer, благодаря чему его положение не зависит от перемещения камеры и положения персонажа. В состав интерфейса входит узел InventoryUI, обеспечивающий отображение панели быстрого доступа и взаимодействие с предметами инвентаря вызовом функции _ready().

Листинг 5. Функция _ready().

```
func _ready():
    $HBoxContainer/Button.pressed.connect(select_item.bind("co
in"))
```

```

        $HBoxContainer/Button2.pressed.connect(select_item.bind("w
heat"))
        $HBoxContainer/Button3.pressed.connect(select_item.bind("c
arrot"))
        inventory.inventory_changed.connect(update_ui)
        update_ui()

```

Листинг 6. Функция `update_ui()`.

```

func update_ui():
    coin_label.text = str(inventory.items["coin"])
    wheat_label.text = str(inventory.items["wheat"])
    carrot_label.text = str(inventory.items["carrot"])

```

Ограничение перемещений игрока по неигровым зонам достигается при помощи узла `StaticBody2D`, который в себе содержит несколько узлов `CollisionPolygon2D`. При столкновении узла коллизии игрока `CollisionShape2D` с одним из этих узлов движение персонажа останавливается и может быть продолжено только в другую сторону. Все неигровые зоны ограничены узлами `CollisionPolygon2D`, включая границу карты.

Структуру этой сцены можно увидеть на диаграмме на рисунке 10.

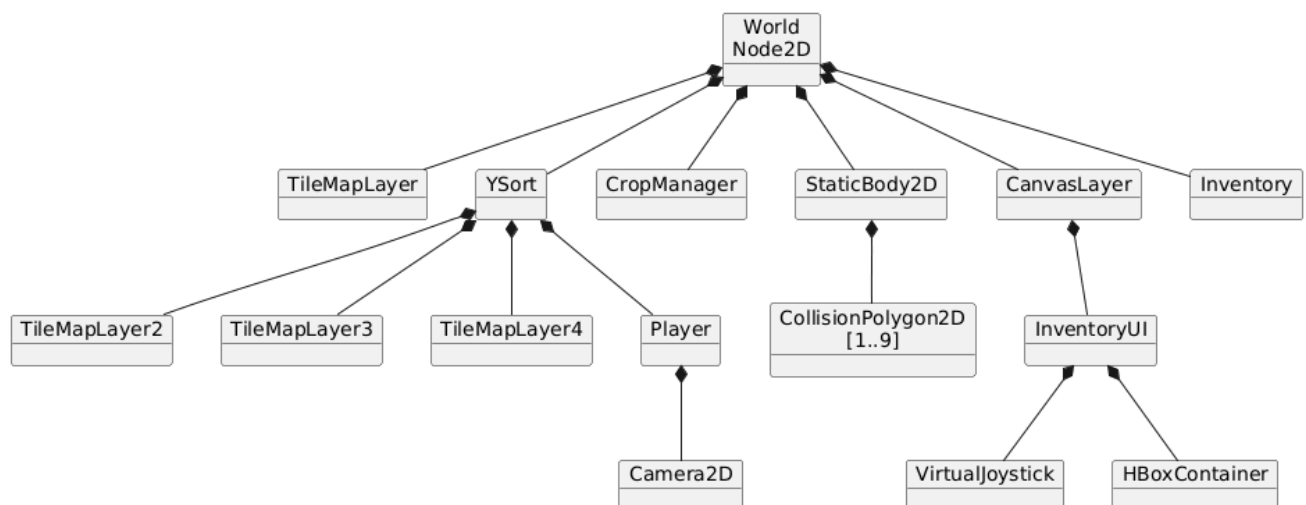


Рисунок 10 – Диаграмма структуры сцены world

Благодаря такой реализации представляется возможность тестировать и расширять приложение. Диаграмма классов приложения представлена на рисунке 11.

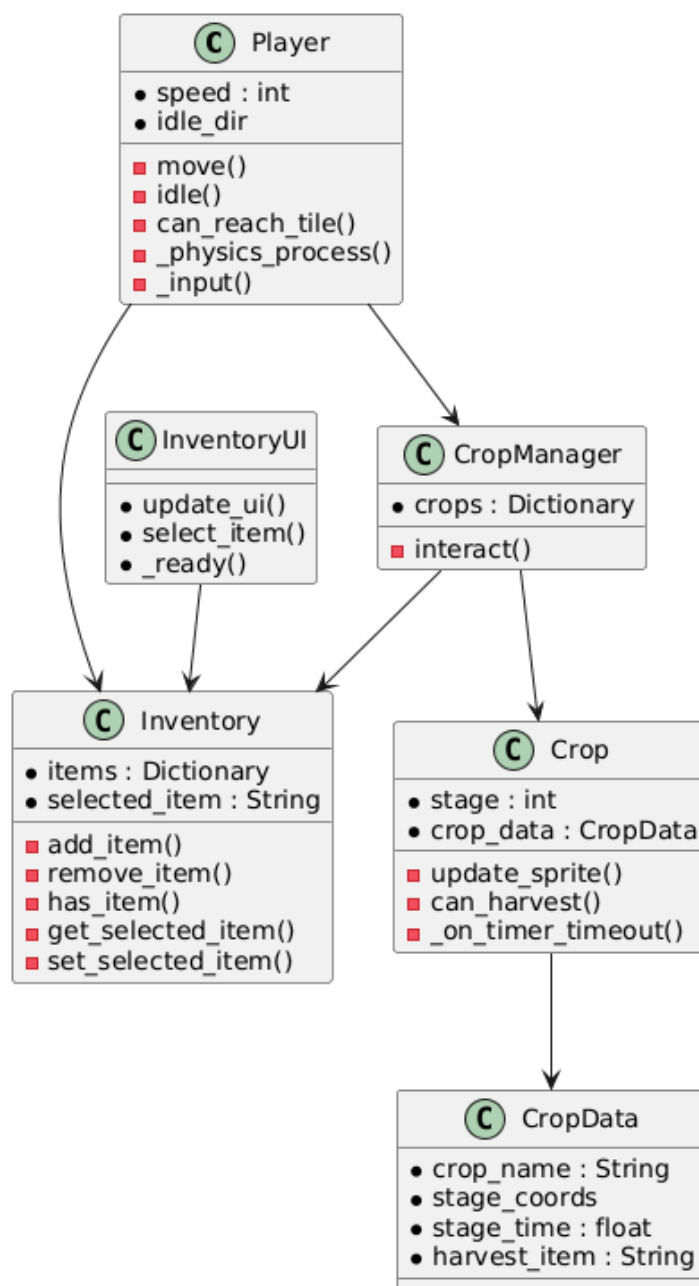


Рисунок 11 – Диаграмма классов приложения

4 Тестирование

После успешной реализации первого прототипа приложения следует заняться его тестированием. Данный этап может помочь выявить ошибки и проблемы в коде, а также улучшить пользовательский опыт взаимодействия, и является одним из важнейших этапов в разработке ПО.

В данном пункте будут протестированы такие программные моменты как:

- передвижение персонажа;
- ограничение передвижения для неигровых зон;
- посадка растений;
- сбор урожая.

4.1 Передвижение персонажа

После запуска игры пользователь может увидеть локацию спавна персонажа и самого персонажа. На данном этапе у пользователя на экране представлен кликабельный инвентарь с начальными ресурсами и виртуальный джойстик, который позволяет управлять персонажем. Это продемонстрировано на рисунке 12.



Рисунок 12 – Стартовая позиция

Изм.	Лист	№докум.	Подпись	Дата

Далее необходимо протестировать корректность передвижения персонажа с помощью джойстика, а также корректность отрисовки анимаций. На рисунках 13 – 16 представлено передвижение персонажа во все стороны



Рисунок 13 – Движение персонажа вправо



Рисунок 14 – Движение персонажа влево



Рисунок 15 – Движение персонажа вниз



Рисунок 16 – Движение персонажа вверх

4.2 Ограничение передвижения для неигровых зон

Есть определённые части карты, в которых нахождение игрока не задумано. А именно река, озеро, скалы, за пределами карты, внутри деревьев и объектов. Для наглядности работы коллизий в дебаггере был включен режим, отрисовывающий их. После того, как пользователь их касается, персонаж перестаёт двигаться. На рисунке 17 была осуществлена попытка попасть внутрь коллизий, а на рисунке 18 была осуществлена попытка выйти за границы карты. Обе попытки оказались неудачными.



Рисунок 17 – Попытка попасть внутрь коллизий



Рисунок 18 – Попытка выхода за границы карты

4.3 Посадка растений

В данном пункте будет рассмотрен один из основных функционалов данного приложения – посадка растений. Чтобы посадить растение, нужно подойти к одной из свободных грядок, выбрать в инвентаре нужную культуру и нажать на грядку. Все стадии роста растения можно увидеть на рисунках 19-22.



Рисунок 19 – Растение посажено



Рисунок 20 – Вторая стадия роста



Рисунок 21 – Третья стадия роста

Изм.	Лист	№докум.	Подпись	Дата

ПГУ 1.090301.34.001 ПЗ

Лист

40



Рисунок 22 – Растение готово к сбору

На скриншотах можно заметить, что после посадки количество пшеницы в инвентаре уменьшилось. Это означает, что не только сама механика посадки работает корректно, но и инвентарь.

После этого нужно выполнить проверку на ошибки. Если выбрать монеты в инвентаре и нажать на грядку, то ничего не происходит. Вывод в консоль можно увидеть на рисунке 23.



Рисунок 23 – Посадка неподходящего предмета в грядку

Если попробовать посадить растение не на грядке, а в другом месте, это тоже не получится сделать. Для тайлов грядок добавлен специальный параметр

CustomDataLayerco значением farmable = 1. У всех остальных клеток он равняется 0. Вывод в консоль после попытки посадить растение не на грядку можно увидеть на рисунке 24.



Рисунок 24 – Посадка растения в неподходящее место

Следующая ситуация, которую необходимо рассмотреть – посадка растения в грядку, в которую уже посажено растение. Данный вариант тоже предусмотрен в программе – перед тем как посадить растение производится считывание информации с выбранного тайла. Если обнаруживается, что тайл занят – посадка растения отклоняется. Вывод в консоль после попытки посадить растение на занятую грядку можно увидеть на рисунке 25.



Рисунок 25 – Посадка растения в занятое место

Последняя ситуация, которую необходимо рассмотреть – попытка посадить растение после того, как оно закончилось в инвентаре. В скрипте `Inventory.gd` есть функция `hasItem()`, которая предназначена специально для такой ситуации. Перед посадкой растения вызывается данная функция для проверки возможности посадки, и если количество растений в инвентаре = 0, то посадка не происходит. Вывод в консоль после попытки посадить растение закончившееся в инвентаре можно увидеть на рисунке 26.

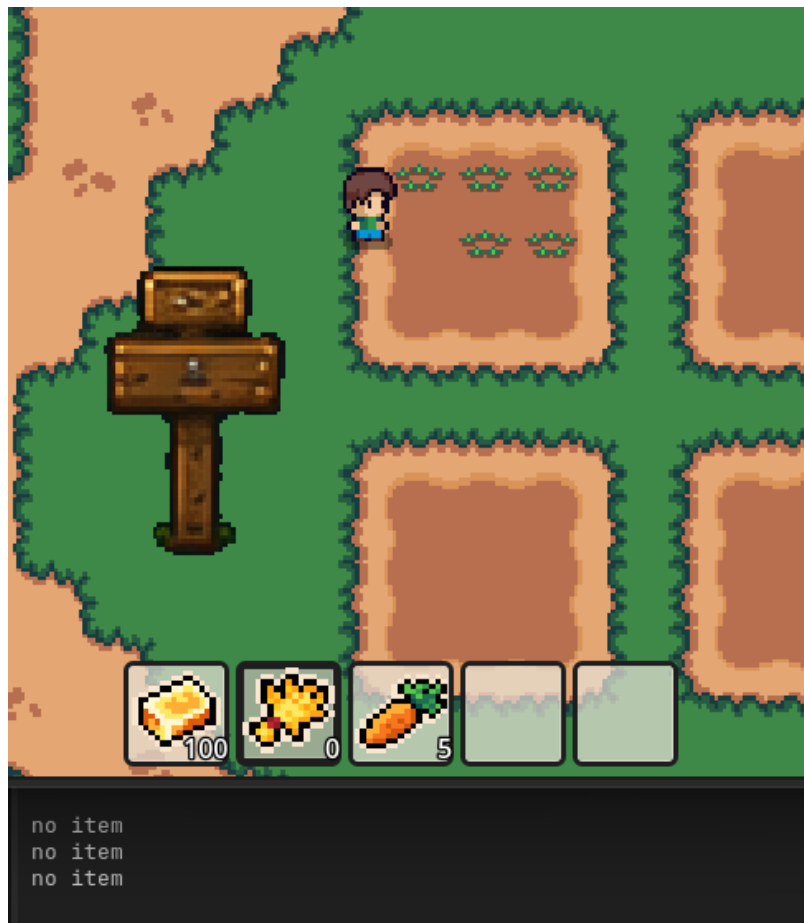


Рисунок 26 – Посадка закончившегося растения

4.4 Сбор урожая

Так же как и посадка растений, сбор урожая является основным функционалом данного приложения. Чтобы собрать урожай нужно подойти к грядке, на которой находится растение на четвёртой стадии роста, и нажать на неё. После этого растение с грядки должно исчезнуть, у персонажа проиграться соответствующая анимация, а количество растений в инвентаре увеличиться. В игре реализовано так, что пользователь получает в два раза больше ресурсов после сбора урожая, чем количество ресурсов, затраченных на посадку. На рисунках 27-28 можно увидеть процесс сбора урожая.



Рисунок 29 – До сбора моркови



Рисунок 30 – После сбора моркови

4.5 Вывод

После создания приложения было выявлено некоторое количество ошибок. Благодаря тестированию ПО получилось обработать и исправить большую часть из них, что позволило сделать более надежное и отказоустойчивое приложение.

					ПГУ 1.090301.34.001 ПЗ	Лист
						47
Изм.	Лист	№докум.	Подпись	Дата		

5 Охрана труда

В современном мире компьютерные технологии являются неотъемлемой частью повседневной жизни человека. Большая часть профессиональной деятельности связана с использованием персональных компьютеров, мобильных устройств и другой вычислительной техники. Длительное взаимодействие с данными устройствами без соблюдения правил безопасности может привести к ухудшению состояния здоровья пользователя, а также к повреждению оборудования.

В данном разделе будут рассмотрены основные правила и требования безопасности при работе с персональным компьютером, которые позволяют снизить влияние вредных факторов на человека.

5.1 Правила и техника безопасности при работе с ПК

При работе с персональным компьютером пользователь может подвергаться воздействию различных опасных и вредных факторов, среди которых можно выделить:

- поражение электрическим током при неисправности оборудования или повреждении кабелей;
- воздействие электромагнитного излучения от монитора и других устройств;
- повышенная нагрузка на зрение;
- неправильное положение тела и нагрузка на опорно-двигательный аппарат;
- повышенный уровень шума от работы компьютерных компонентов;
- возможность возникновения пожара при нарушении правил эксплуатации техники.

Для уменьшения влияния данных факторов необходимо соблюдать установленные правила безопасности при организации рабочего места и эксплуатации компьютерной техники.

В первую очередь рабочее место пользователя должно быть правильно организовано. Помещение должно иметь достаточное освещение, которое не создает бликов на экране монитора и не направлено непосредственно в глаза человеку. Желательно использовать сочетание естественного и искусственного освещения, при этом яркость света не должна вызывать дискомфорт.

Особое внимание следует уделять выбору мебели. Рабочее кресло должно быть удобным и иметь возможность регулировки высоты и положения спинки. Во время работы ноги должны полностью находиться на полу, а спина должна опираться на спинку кресла. Такое положение помогает снизить нагрузку на позвоночник и мышцы человека.

Монитор необходимо устанавливать перед пользователем на расстоянии примерно 50–70 сантиметров от глаз. Верхняя часть экрана должна находиться примерно на уровне глаз или немного ниже. Клавиатура и мышь должны располагаться таким образом, чтобы руки находились в естественном положении и не испытывали лишнего напряжения.

При работе с компьютером запрещается использовать оборудование, имеющее внешние повреждения, неисправные кабели или другие признаки неполадок. Не следует самостоятельно разбирать системный блок, монитор и другие устройства, так как это может привести к поражению электрическим током или повреждению техники.

Для предотвращения перегрева и возникновения пожара необходимо обеспечивать свободную циркуляцию воздуха вокруг системного блока и не накрывать вентиляционные отверстия. Также не рекомендуется оставлять включенный компьютер без присмотра на длительное время.

Для снижения нагрузки на организм необходимо соблюдать режим работы и отдыха. При длительной работе за компьютером следует делать регулярные

перерывы, выполнять упражнения для глаз, рук и мышц спины. Это позволяет снизить усталость и предотвратить развитие заболеваний, связанных с малоподвижным образом жизни.

Соблюдение правил техники безопасности при работе с ПК позволяет сохранить здоровье пользователя, повысить эффективность работы и продлить срок службы компьютерного оборудования.

5.2 Правила и техника безопасности при работе со смартфоном

В настоящее время мобильные устройства широко используются человеком для работы, обучения и повседневного общения. Несмотря на компактные размеры, смартфон также требует соблюдения определенных правил безопасного использования.

При работе со смартфоном необходимо избегать механических повреждений устройства. Не рекомендуется самостоятельно разбирать телефон, изменять его конструкцию или использовать поврежденные аккумуляторы и зарядные устройства. Использование неисправной техники может привести к короткому замыканию, перегреву или возгоранию.

Во время зарядки необходимо использовать исправные зарядные устройства и аксессуары, предназначенные для данного устройства. Не следует оставлять телефон на зарядке в местах с высокой температурой или под прямыми солнечными лучами.

Смартфоны используют беспроводные технологии передачи данных, поэтому устройство создает радиочастотное излучение. Для уменьшения времени воздействия рекомендуется использовать гарнитуру при длительных разговорах и не держать телефон рядом с телом без необходимости.

Также следует учитывать влияние длительного использования смартфона на здоровье человека. Постоянное использование мобильного устройства может привести к повышенной нагрузке на зрение, нарушение режима сна и снижение

					ПГУ 1.090301.34.001 ПЗ	Лист
						50
Изм.	Лист	№докум.	Подпись	Дата		

концентрации внимания. Для предотвращения таких последствий рекомендуется ограничивать время непрерывного использования телефона и делать перерывы.

Таким образом, соблюдение правил безопасности при работе с компьютером и мобильными устройствами помогает снизить возможные риски для здоровья человека и обеспечивает безопасное использование современных технологий.

					ПГУ 1.090301.34.001 ПЗ	Лист
						51
Изм.	Лист	№докум.	Подпись	Дата		

Заключение

Результатом выполнения данной выпускной квалификационной работы можно считать успешно разработанную мобильную 2D-игру, полностью соответствующую заданию. Также были выполнены следующие задачи:

- полный анализ темы и аналогов;
- выбор инструментов разработки;
- анализ архитектуры разрабатываемого приложения;
- непосредственная разработка приложения;
- тестирование ПО.

В результате было создано полностью протестированное приложение для мобильных устройств, написанное на языке программирования GDScript. Также в ходе разработки были получены навыки работы с игровым движком Godot. Таким образом, результаты показали, что приложение полностью соответствует всем поставленным задачам, имеет удобный пользовательский интерфейс и выполняет свою работу корректно.

В перспективе приложение можно будет улучшить, добавив новый функционал и новые игровые механики, а также доработать пользовательский интерфейс для более комфортного использования со стороны пользователей. Данное мобильное приложение готово для использования в сфере развлечений и мобильных игр.

Список использованных источников

1. godotengine.org: [Электронный ресурс]. URL: <https://godotengine.org/> (дата обращения 15.04.2026)
2. ru.wikipedia.org: [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/Godot> (дата обращения 15.04.2026)
3. gdscrip.com: [Электронный ресурс]. URL: <https://gdscrip.com/tutorials/> (дата обращения 16.04.2026)
4. sky.pro: [Электронный ресурс]. URL: <https://sky.pro/wiki/gamedev/sozдание-2d-igr-na-godot-poshagovoe-rukovodstvo/> (дата обращения 20.04.2026)
5. itch.io: [Электронный ресурс]. URL: <https://kenmi-art.itch.io/cute-fantasy-rpg> (дата обращения 23.04.2026)
6. ohrana-bez.ru: [Электронный ресурс]. URL: <https://oxrana-bez.ru/stati/tehnika-bezopasnosti-pri-rabote-s-personalnym-kompyuterom/> (дата обращения 17.05.2026)
7. garant.ru: [Электронный ресурс]. URL: <https://www.garant.ru/products/ipo/prime/doc/409555249/> (дата обращения 17.05.2026)

Приложение А

(обязательное)

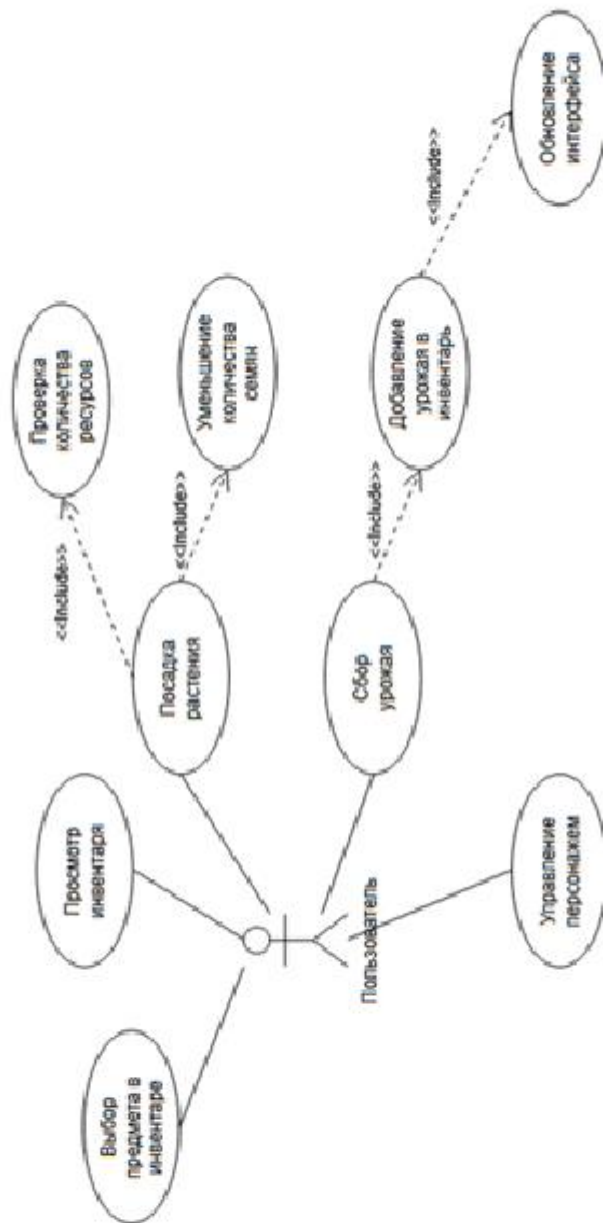
Мобильная 2D-игра для устройств на ОСAndroid

UML-диаграммы

					ПГУ 1.090301.34.001 ПЗ	Лист
						54
Изм.	Лист	№докум.	Подпись	Дата		

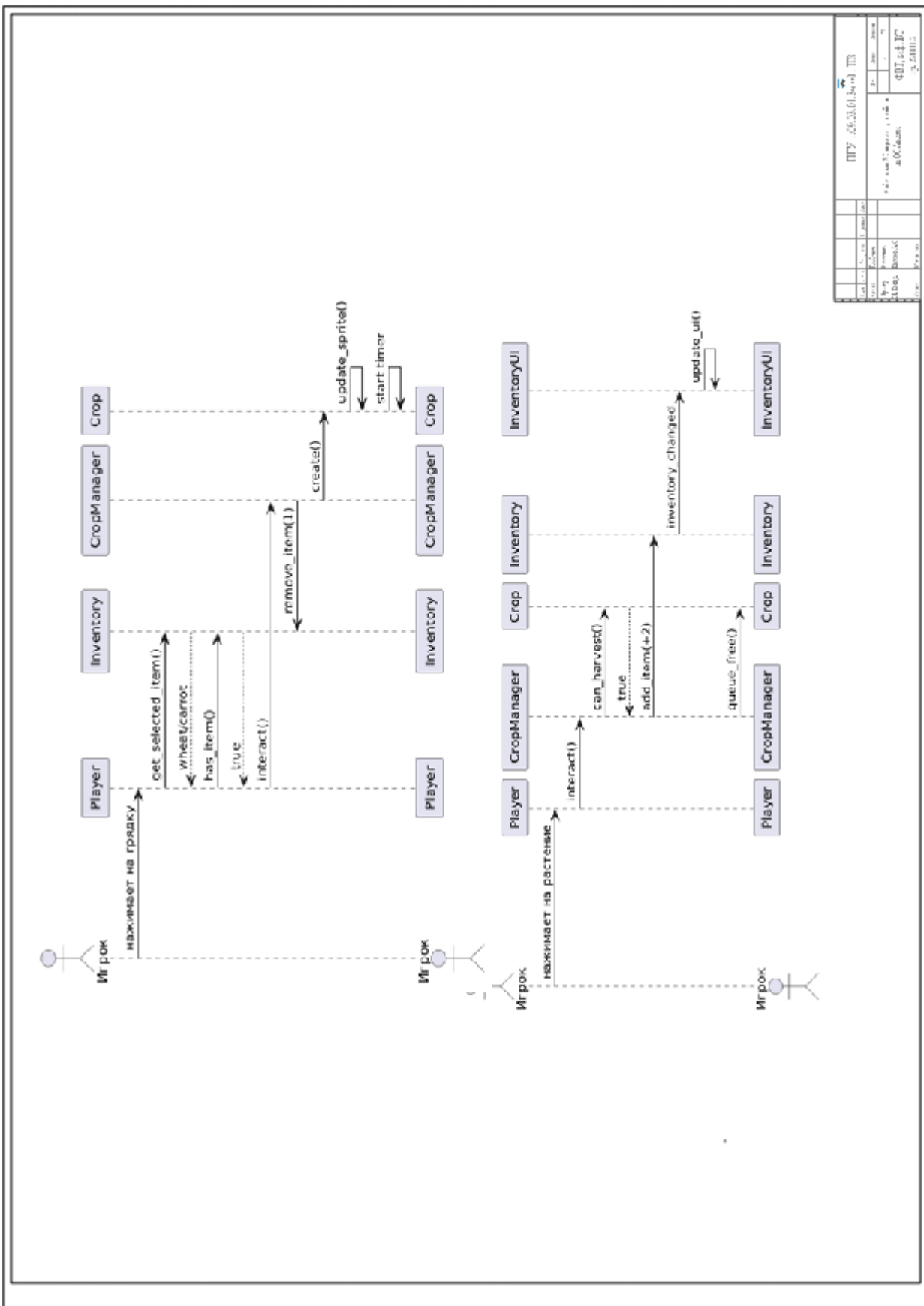
UML-диаграмма прецедентов

Диаграмма вариантов использования

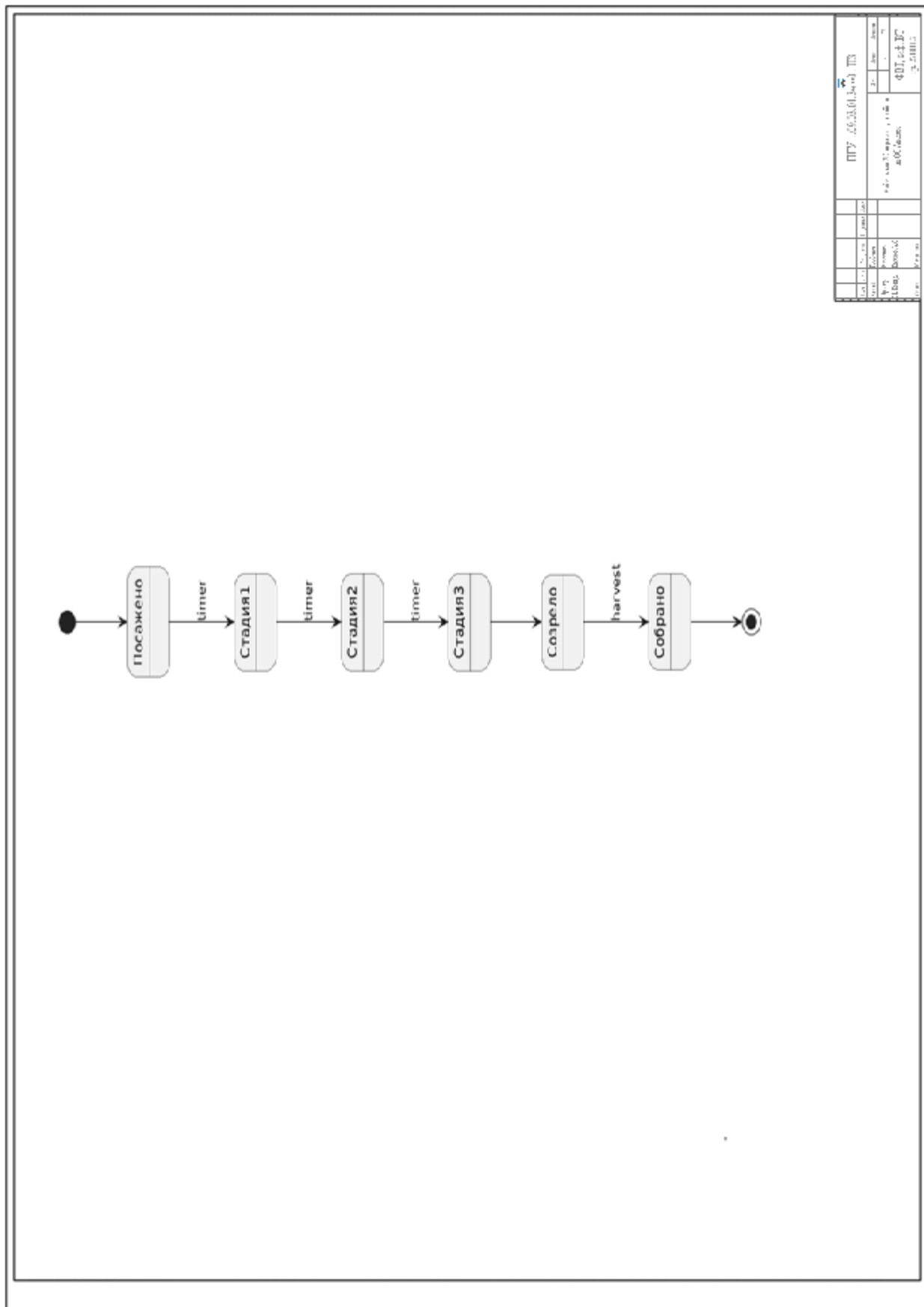


Имя	№ докум.	Исполн.	Дата	Стр.
ПГУ 1.090301.34.001 ПЗ				
Имя	№ докум.	Исполн.	Дата	Стр.
ПГУ 1.090301.34.001 ПЗ				
Имя	№ докум.	Исполн.	Дата	Стр.
ПГУ 1.090301.34.001 ПЗ				

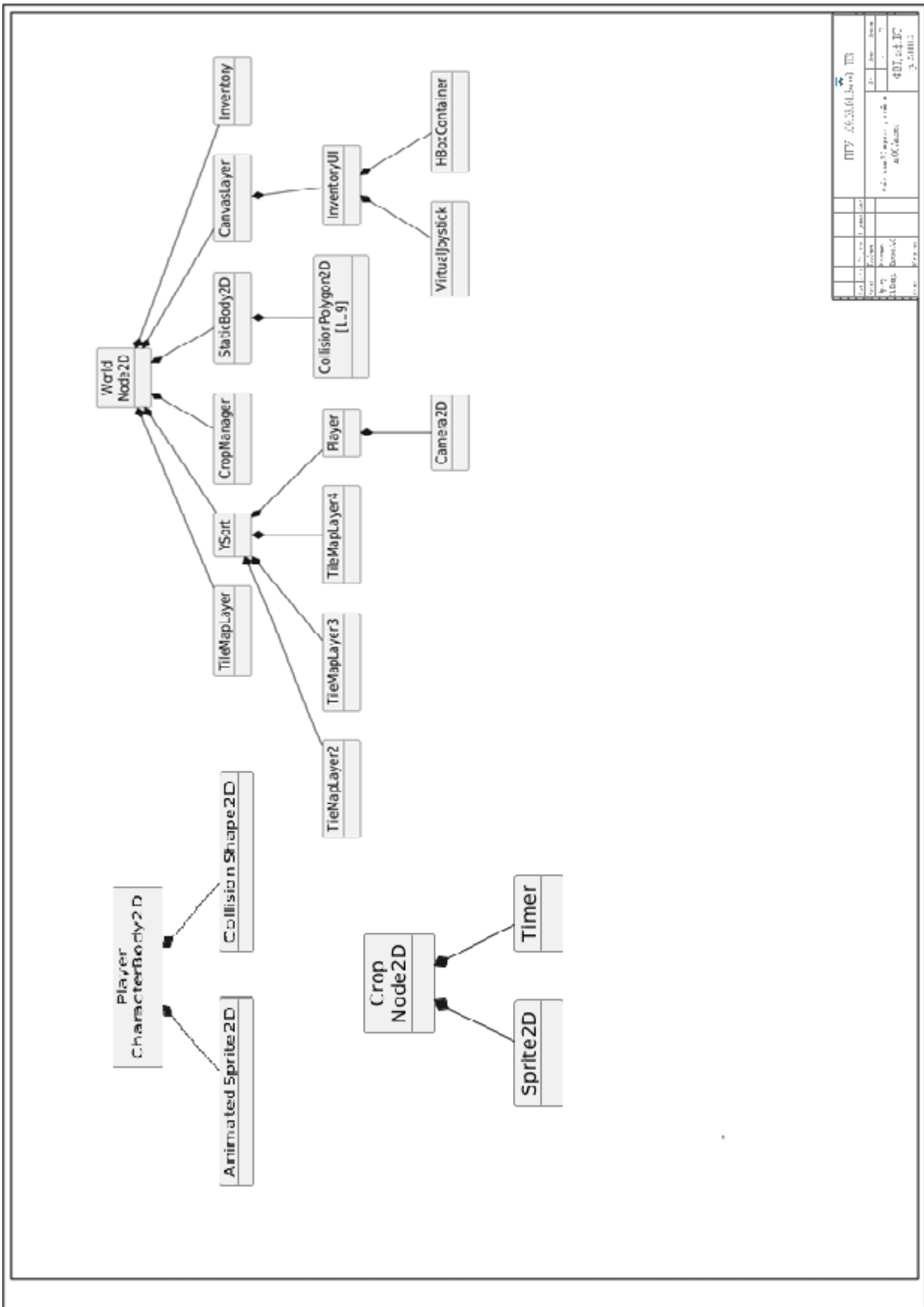
UML-диаграмма деятельности



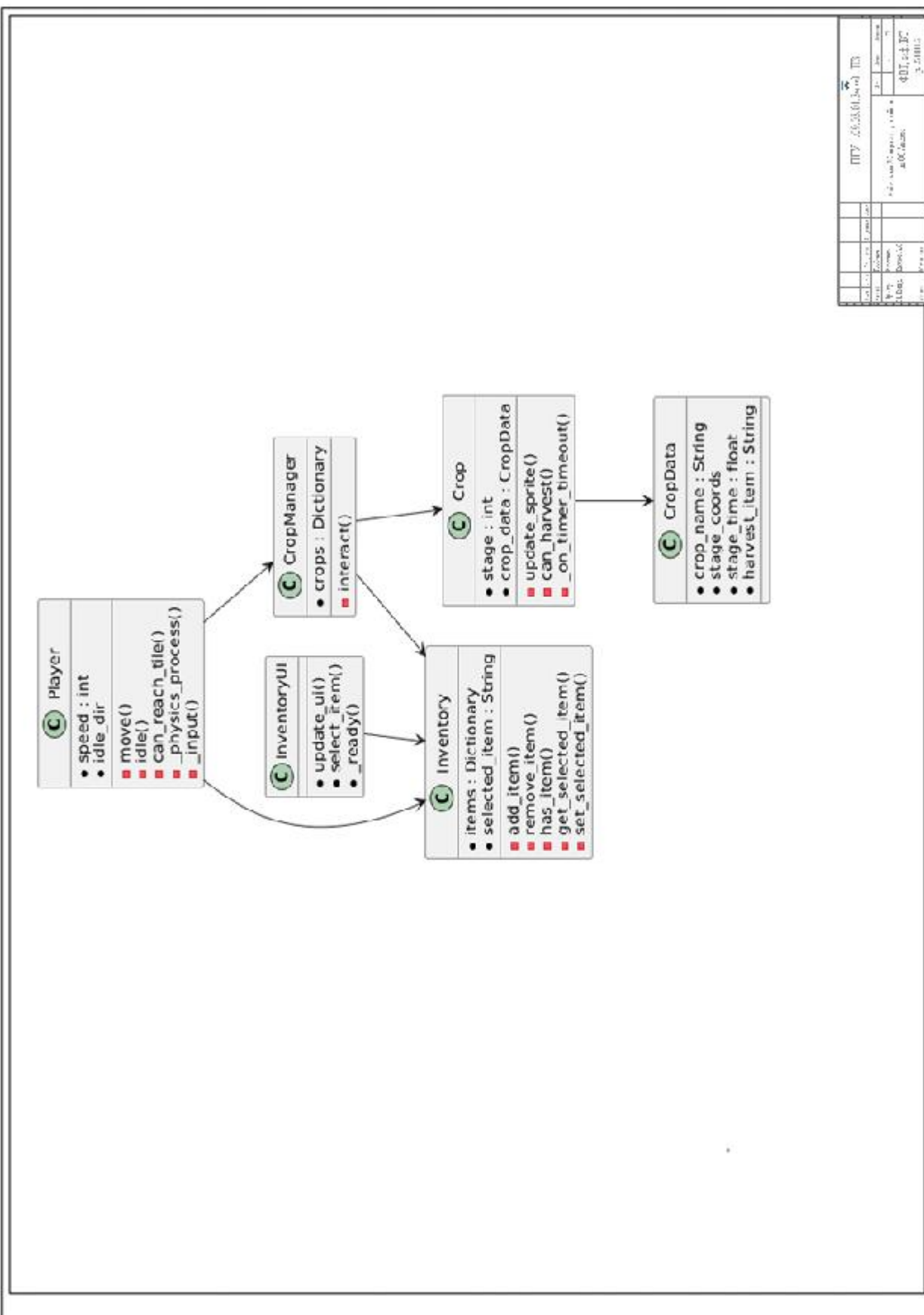
UML-диаграмма последовательностей



UML-диаграмма развертывания



UML-диаграмма классов



Приложение Б

(обязательное)

Мобильная 2D-игра для устройств на ОСAndroid

Листинг программы

					ПГУ 1.090301.34.001 ПЗ	Лист
						60
Изм.	Лист	№докум.	Подпись	Дата		

Файл «player.gd»

```
extends CharacterBody2D

enum {
    DOWN,
    UP,
    LEFT,
    RIGHT
}

@onready var crop_manager = $"../..../CropManager"
@onready var farm_layer = $"../..../TileMapLayer"
@export var carrot_data: CropData
@export var wheat_data: CropData
@onready var inventory = $"../..../Inventory"
@onready var anim = $AnimatedSprite2D
var speed = 100
var idle_dir = DOWN

func _physics_process(delta: float) -> void:
    var direction = Input.get_vector(
        "left",
        "right",
        "up",
        "down"
    )

    if direction != Vector2.ZERO:
        move(direction)
    else:
        idle()

    move_and_slide()

func move(direction: Vector2):
    velocity = direction * speed

    # вверх/вниз
    if abs(direction.y) > abs(direction.x):
        if direction.y < 0:
            anim.play("up")
            idle_dir = UP
        else:
            anim.play("down")
            idle_dir = DOWN

    # лево/право
    else:
        if direction.x < 0:
            anim.flip_h = true
            anim.play("front")
            idle_dir = LEFT
```

```

        else:
            anim.flip_h = false
            anim.play("front")
            idle_dir = RIGHT

func idle():
    velocity.x = 0
    velocity.y = 0
    if velocity.x == 0 and velocity.y == 0:
        match idle_dir:
            DOWN:
                anim.play("idle_down")
            UP:
                anim.play("idle_up")
            LEFT:
                anim.flip_h = true
                anim.play("idle_front")
            RIGHT:
                anim.flip_h = false
                anim.play("idle_front")

func can_reach_tile(tile_position: Vector2i) -> bool:
    # Получаем локальную позицию клетки относительно TileMapLayer
    var tile_local_position =
        farm_layer.map_to_local(tile_position)

    # Переводим её в глобальные координаты
    var tile_global_position =
        farm_layer.to_global(tile_local_position)

    # Проверяем расстояние от игрока до клетки
    return global_position.distance_to(tile_global_position) <= 24

func _input(event):
    if event.is_action_pressed("left_click"):

        var world_position = get_global_mouse_position()
        var tile_position = farm_layer.local_to_map(
            farm_layer.to_local(world_position)
        )

        if can_reach_tile(tile_position):

            match inventory.get_selected_item():

                "wheat":
                    if inventory.has_item("wheat"):
                        crop_manager.interact(tile_position,
wheat_data)

```

```

        "carrot":
            if inventory.has_item("carrot"):
                crop_manager.interact(tile_position,
carrot_data)
        "coin":
            print("no item")

```

Файл «inventory.gd»

```
extends Node
```

```
signal inventory_changed
```

```

var items = {
"coin": 100,
"wheat": 5,
"carrot": 5
}

```

```
var selected_item = "coin"
```

```

funcset_selected_item(item_name):
    selected_item = item_name

```

```

funcget_selected_item():
    return selected_item

```

```

funcchas_item(item_name):
    return items.get(item_name, 0) > 0

```

```
funcadd_item(item_name, amount):
```

```
    items[item_name] = items.get(item_name, 0) + amount
```

```
    inventory_changed.emit()
```

```
funcremove_item(item_name, amount):
```

```
    if items.get(item_name, 0) >= amount:
```

```
        items[item_name] -= amount
```

```
        inventory_changed.emit()
```

Файл «inventory_ui.gd»

```
extends Control
```

```
@onready var inventory = $"../../Inventory"
```

```
@onready var coin_label = $HBoxContainer/Button/Label
```

					ПГУ 1.090301.34.001 ПЗ	Лист
						63
Изм.	Лист	№докум.	Подпись	Дата		

```

@onready var wheat_label = $HBoxContainer/Button2/Label
@onready var carrot_label = $HBoxContainer/Button3/Label

func select_item(item_name):
    inventory.set_selected_item(item_name)

func _ready():
    $HBoxContainer/Button.pressed.connect(select_item.bind("coin"))
    $HBoxContainer/Button2.pressed.connect(select_item.bind("wheat"))
    $HBoxContainer/Button3.pressed.connect(select_item.bind("carrot"))
    inventory.inventory_changed.connect(update_ui)
    update_ui()

func update_ui():

    coin_label.text = str(inventory.items["coin"])
    wheat_label.text = str(inventory.items["wheat"])
    carrot_label.text = str(inventory.items["carrot"])

```

Файл «crops.gd»

```

extends Node2D

@export var crop_data: CropData
@export var atlas_texture: Texture2D

@onready var sprite = $Sprite2D
@onready var timer = $Timer

var stage = 0

func _ready():
    timer.timeout.connect(_on_timer_timeout)

    update_sprite()

    timer.wait_time = crop_data.stage_time

    timer.start()

func update_sprite():

    var coord = crop_data.stage_coords[stage]

    var texture = AtlasTexture.new()

```

```

texture.atlas = atlas_texture

texture.region = Rect2(
    coord.x * 16,
    coord.y * 16,
    16,
    16
)

sprite.texture = texture

func _on_timer_timeout():

    if stage < crop_data.stage_coords.size() - 1:

        stage += 1

        update_sprite()

        timer.start()

func can_harvest() -> bool:

    return stage == crop_data.stage_coords.size() - 1

```

Файл «crops_data.gd»

```

extends Resource
class_name CropData

@export var crop_name: String

@export var stage_coords: Array[Vector2i]

@export var stage_time: float = 30

@export var harvest_item: String

```

Файл «crop_manager.gd»

```

extends Node2D

@onready var inventory = $"../Inventory"
@export var crop_scene: PackedScene
@export var atlas_texture: Texture2D
@export var farm_layer: TileMapLayer

var crops = {}

```

```

func interact(tile_position: Vector2i, crop_data: CropData):
    if tile_position in crops:

        var crop = crops[tile_position]

        if crop.can_harvest():
            inventory.add_item( crop.crop_data.harvest_item, 2 )
            print("Получен предмет:", crop.crop_data.harvest_item)

            crops.erase(tile_position)

            crop.queue_free()

        return

var tile_data = farm_layer.get_cell_tile_data(tile_position)
var value = tile_data.get_custom_data("farmable")
if tile_data == null:
    return
if value != 1:
    return
#if !tile_data.get_custom_data("farmable"):
#    return
inventory.remove_item(crop_data.harvest_item, 1)
var crop = crop_scene.instantiate()

crop.crop_data = crop_data
crop.atlas_texture = atlas_texture

get_parent().get_node("YSort").add_child(crop)

crop.global_position = farm_layer.to_global(
    farm_layer.map_to_local(tile_position)
) + Vector2(0, -6)

crops[tile_position] = crop

```

Приложение В

(обязательное)

Мобильная 2D-игра для устройств на ОСAndroid

Презентация

					ПГУ 1.090301.34.001 ПЗ	Лист
						67
Изм.	Лист	№докум.	Подпись	Дата		

Мобильная 2D-игра для устройств на ОС Android

Автор работы: студент группы 22ВВП2 Горбунов Д.А.

Научный руководитель: доцент кафедры ВТ Кузнецов А.В.

Пенза, 2026

Актуальность проекта

Автор работы: студент группы 22ВВП2 Горбунов Д.А.

Научный руководитель: доцент кафедры ВТ Кузнецов А.В.

					ПГУ 1.090301.34.001 ПЗ	Лист
						68
Изм.	Лист	№докум.	Подпись	Дата		

Цели работы

Автор работы: студент группы 22ВВП2 Горбунов Д.А.

Научный руководитель: доцент кафедры ВТ Кузнецов А.В.

Обзор аналогов

Автор работы: студент группы 22ВВП2 Горбунов Д.А.

Научный руководитель: доцент кафедры ВТ Кузнецов А.В.

Обзор аналогов

Автор работы: студент группы 22ВВП2 Горбунов Д.А.

Научный руководитель: доцент кафедры ВТ Кузнецов А.В.

Игровая карта

Автор работы: студент группы 22ВВП2 Горбунов Д.А.

Научный руководитель: доцент кафедры ВТ Кузнецов А.В.

					ПГУ 1.090301.34.001 ПЗ	Лист
						70
Изм.	Лист	№докум.	Подпись	Дата		

Игровой интерфейс

Автор работы: студент группы 22ВВП2 Горбунов Д.А.

Научный руководитель: доцент кафедры ВТ Кузнецов А.В.

Заключение

Автор работы: студент группы 22ВВП2 Горбунов Д.А.

Научный руководитель: доцент кафедры ВТ Кузнецов А.В.

					ПГУ 1.090301.34.001 ПЗ	Лист
						71
Изм.	Лист	№докум.	Подпись	Дата		

Спасибо за внимание!