



**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение
высшего образования

«ИРКУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ФГБОУ ВО «ИГУ»

Факультет бизнес-коммуникаций и информатики
Кафедра естественнонаучных дисциплин

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине Б1.В.20 Разработка компьютерных игр

направление подготовки 09.03.03 Прикладная информатика

направленность (профиль) Прикладная информатика (разработка программного обеспечения)

Одобрено
УМК факультета бизнес-коммуникаций
и информатики

Разработан в соответствии с ФГОС ВО

с учетом требований проф. стандарта

Председатель УМК

М.Г. Синчурина
ФИО, должность, ученая степень, звание

подпись, печать

Разработчики:


(подпись)

старший преподаватель
(занимаемая должность)

А.В. Киселев
(инициалы, фамилия)

Цель фонда оценочных средств. Оценочные средства предназначены для контроля и оценки образовательных достижений обучающихся, освоивших программу учебной дисциплины «Б1.В.20 Разработка компьютерных игр». Перечень видов оценочных средств соответствует рабочей программе дисциплины.

Фонд оценочных средств включает контрольные материалы для проведения текущего контроля (в следующих формах: тест, практическое задание) и промежуточной аттестации в форме вопросов и заданий к экзамену.

Структура и содержание заданий – задания разработаны в соответствии с рабочей программой дисциплины «Б1.В.20 Разработка компьютерных игр».

1. Паспорт фонда оценочных средств

Компетенция	Индикаторы компетенций	Результаты обучения
ПК-1 ППК-Р1 Способен разрабатывать и отлаживать программный код	ПК-1 ППК-Р1.1 Выполняет формализацию и алгоритмизацию поставленных задач для разработки программного кода	• ПК-1.1 ППК-Р1.1. 3-3. Знает методы и приемы алгоритмизации поставленных задач. • ПК-1.1 ППК-Р1.1. У-2. Умеет применять алгоритмы решения типовых задач в соответствующих областях.
	ПК-1 ППК-Р1.2 Формализует задачу ИТ отрасли в язык естественнонаучных дисциплин	• ПК-1.2 ППК-Р1.4 3-1. Знает синтаксис выбранного языка программирования, особенности программирования на этом языке, стандартные библиотеки языка программирования. • ПК-1.2 ППК-Р1.4 3-3. Знает технологии программирования. • ПК-1.2 ППК-Р1.4 У-2. Умеет использовать выбранную среду программирования. • ПК-1.2 ППК-Р1.4 У-3. Умеет использовать возможности имеющейся технической и/или программной архитектуры для написания программного кода.
	ПК-1 ППК-Р1.3 Осуществляет обоснованный выбор методов и алгоритмов для программной реализации формальной математической модели	• ПК-1.7 ППК-Р1.3. У-1. Умеет проводить сравнительный анализ и обоснование выбора алгоритмов для программной реализации модели. • ПК-1.7 ППК-Р1.3. В-2. Владеет навыками оценки эффективности выбранных алгоритмов.
	ПК-1 ППК-Р1.4 Разрабатывает программный код с использованием языков программирования	• ПК-1.2 ППК-Р1.4 3-1. Знает синтаксис выбранного языка программирования, особенности программирования на этом языке, стандартные библиотеки языка программирования. • ПК-1.2 ППК-Р1.4 3-3. Знает технологии программирования. • ПК-1.2 ППК-Р1.4 У-2. Умеет использовать выбранную среду программирования. • ПК-1.2 ППК-Р1.4 У-3. Умеет использовать возможности имеющейся технической и/или программной архитектуры для написания программного кода.
	ПК-1 ППК-Р1.5 Оформляет программный код в соответствии с установленными требованиями	• ПК-1.3 ППК-Р1.5 3-2. Знает основные стандарты оформления технической документации на компьютерное программное обеспечение. • ПК-1.3 ППК-Р1.5 У-3. Умеет применять инструментарий для создания и актуализации исходных текстов программ.
	ПК-1 ППК-Р1.6 Работает с системой управления версиями программного кода	• ПК-1.4 ППК-Р1.6. 3-1. Знает возможности используемой системы управления версиями и вспомогательных инструментальных программных средств. • ПК-1.4 ППК-Р1.6. У-1. Умеет регистрировать изменения исходного текста программного кода в системе управления версиями. • ПК-1.4 ППК-Р1.6. У-3. Умеет выполнять слияние, разделение и сравнение исходных текстов программного кода.
	ПК-1 ППК-Р1.7 Проверяет и отлаживает программный код	• ПК-1.5 ППК-Р1.7 3-1. Знает методы и приемы отладки программного кода. • ПК-1.5 ППК-Р1.7 У-1. Умеет выявлять ошибки в программном коде. • ПК-1.5 ППК-Р1.7 У-2. Умеет отлаживать программный код на уровне программных модулей.

Компетенция	Индикаторы компетенций	Результаты обучения
ПК-2 ППК-Р2 Способен проверять работоспособность и проводить рефакторинг кода программного обеспечения	ПК-2 ППК-Р2.1 Разрабатывает тестовые наборы данных для проверки работоспособности компьютерного программного обеспечения	• ПК-2.1 ППК-Р2.1 З-1. Знает методы создания и документирования контрольных примеров и тестовых наборов данных. • ПК-2.1 ППК-Р2.1 У-1. Умеет разрабатывать и оформлять контрольные примеры для проверки работоспособности компьютерного программного обеспечения.
	ПК-2 ППК-Р2.2 Проверяет работоспособность компьютерного программного обеспечения	• ПК-2.2 ППК-Р2.2 З-1. Знает методы и средства проверки работоспособности компьютерного программного обеспечения. • ПК-2.2 ППК-Р2.2 У-1. Умеет применять методы и средства проверки работоспособности компьютерного программного обеспечения.
	ПК-2 ППК-Р2.3 Исправляет дефекты программного кода, зафиксированных в базе данных дефектов	• ПК-2.3 ППК-Р2.3 З-1. Знает типичные ошибки, возникающие при разработке компьютерного программного обеспечения, методы их диагностики и исправления. • ПК-2.3 ППК-Р2.3 З-2. Знает методы и приемы отладки программного кода. • ПК-2.3 ППК-Р2.3 У-2. Умеет выяснять причины возникновения дефектов программного кода. • ПК-2.3 ППК-Р2.3 У-3. Умеет вносить изменений в программный код для устранения выявленных дефектов.
	ПК-2 ППК-Р2.4 Выполняет рефакторинг и инспекцию программного кода (код-ревью)	• ПК-2.4 ППК-Р2.4 З-1. Знает методы и средства рефакторинга и инспекции программного кода. • ПК-2.4 ППК-Р2.4 У-1. Умеет анализировать программный код на соответствие требованиям по читаемости и производительности. • ПК-2.4 ППК-Р2.4 У-2. Умеет проводить инспекцию программного кода для поиска не обнаруженных на ранних стадиях разработки компьютерного программного обеспечения ошибок и критических мест.
ПК-6 ИП-2 Способен оптимизировать производительность программного обеспечения	ПК-6 ИП-2.1 Выполняет мониторинг производительности программного обеспечения	• ПК-6.1 ИП-2.1 З-1. Знает методы и средства мониторинга производительности компьютерного программного обеспечения. • ПК-6.1 ИП-2.1 У-1. Умеет применять методы и средства мониторинга производительности компьютерного программного обеспечения.
	ПК-6 ИП-2.2 Выполняет оптимизацию программного кода	• ПК-6.2 ИП-2.2 З-1. Знает методы и средства оптимизации производительности компьютерного программного обеспечения. • ПК-6.2 ИП-2.2 У-2. Умеет вырабатывать варианты оптимизации производительности компьютерного программного обеспечения.

Компетенция	Индикаторы компетенций	Результаты обучения
ПК-8 ППК-Р7 Способен применять искусственный интеллект (ИИ) для генерации и отладки программного кода	ПК-8 ППК-Р7.1 Применяет ИИ-инструменты для генерации программного кода	• ПК-8.1 ППК-Р7.1 З-1. Знает принципы работы современных генеративных ИИ-моделей для генерации кода. • ПК-8.1 ППК-Р7.1 У-1. Умеет формулировать корректные текстовые запросы (промты) для генерации кода.
	ПК-8 ППК-Р7.2 Использует ИИ для анализа и отладки кода	• ПК-8.2 ППК-Р7.2 З-1. Знает методы ИИ-анализа кода. • ПК-8.2 ППК-Р7.2 У-2. Умеет интерпретировать рекомендации ИИ по исправлению кода.
	ПК-8 ППК-Р7.3 Оптимизирует код с помощью ИИ	• ПК-8.3 ППК-Р7.3 З-1. Знает методы ИИ-оптимизации. • ПК-8.3 ППК-Р7.3 У-1. Умеет использовать ИИ для рефакторинга.
	ПК-8 ППК-Р7.4 Оценивает этические и профессиональные аспекты применения ИИ в разработке	• ПК-8.4 ППК-Р7.4 З-1. Знает этические нормы использования ИИ (конфиденциальность, плагиат кода и т.п.). • ПК-8.4 ППК-Р7.4 У-1. Умеет проверять код на соответствие стандартам после ИИ-генерации.

2. Показатели и критерии оценивания компетенций, шкалы оценивания

2.1. Показатели и критерии оценивания компетенций

№ п/п	Раздел, тема	Код индикатора компетенции	Наименование ОС	
			ТК	ПА
1	Компьютерные игры и разработка	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3	Тест	Тест
2	Графический движок Godot	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.2, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7	Тест, Пз	Тест
3	Средства разработки. Основы языка программирования GDScript	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5	Тест, Пз	Тест
4	Событийное программирование в Godot. Основные методы и их назначение	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4	Тест, Пз	Тест
5	Работа с графическими примитивами. Координатные системы: трансформация	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3, ПК-2 ППК-Р2.2	Тест, Пз	Тест

№ п/п	Раздел, тема	Код индикатора компетенции	Наименование ОС	
			ТК	ПА
6	Работа с аудио и графикой	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.2, ПК-8 ППК-Р7.1, ПК-8 ППК-Р7.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-8 ППК-Р7.3, ПК-8 ППК-Р7.4	Тест, Пз	Тест
7	Векторная математика. Силы и движение	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7	Тест, Пз	Тест
8	Системы частиц	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.2, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-8 ППК-Р7.1, ПК-8 ППК-Р7.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-8 ППК-Р7.3, ПК-8 ППК-Р7.4	Тест, Пз	Тест
9	Анимация	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.2, ПК-8 ППК-Р7.1, ПК-8 ППК-Р7.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-8 ППК-Р7.3, ПК-8 ППК-Р7.4	Тест, Пз	Тест
10	Создание многопользовательских игр	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.2, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4	Тест, Пз	Тест
11	Автономные агенты	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.2, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-8 ППК-Р7.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-8 ППК-Р7.3	Тест, Пз	Тест

№ п/п	Раздел, тема	Код индикатора компетенции	Наименование ОС	
			ТК	ПА
12	Генерация карт	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.2, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-8 ППК-Р7.1, ПК-8 ППК-Р7.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-8 ППК-Р7.3, ПК-8 ППК-Р7.4	Тест, Пз	Тест

2.2. Критерии оценивания результатов обучения для текущего контроля успеваемости и промежуточной аттестации

Оценочное средство	Критерии оценивания	Шкала оценивания
Тест	Студентом даны правильные ответы на 91-100% заданий	Отлично
	Студентом даны правильные ответы на 81-90% заданий	Хорошо
	Студентом даны правильные ответы на 71-80% заданий	Удовлетворительно
	Студентом даны правильные ответы менее чем на 70% заданий	Неудовлетворительно
Практическое задание	Задание выполнено верно. Выбран оптимальный путь решения. Присутствует развернутое описание алгоритма решения	Зачтено
	В работе получен неверный ответ, связанный с грубыми ошибками допущенными в ходе решения, либо решение отсутствует полностью	Незачтено

2.3. Оценочные средства для текущего контроля (примеры)

2.3.1. Материалы для компьютерного тестирования обучающихся

Общие критерии оценивания

Процент правильных ответов	Оценка
91% – 100%	5 (отлично)
81% – 90%	4 (хорошо)
71% – 80%	3 (удовлетворительно)
Менее 70%	2 (неудовлетворительно)

Соответствие вопросов теста индикаторам формируемых и оцениваемых компетенций

№ вопроса в тесте	Код индикатора компетенции
1	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
2	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
3	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
4	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
5	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
6	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
7	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
8	ПК-2 ППК-Р2.1
9	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
10	ПК-2 ППК-Р2.1
11	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
12	ПК-2 ППК-Р2.1
13	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
14	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
15	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
16	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
17	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
18	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
19	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
20	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
21	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
22	ПК-2 ППК-Р2.1
23	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3
24	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
25	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
26	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3
27	ПК-2 ППК-Р2.1
28	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
29	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3

Ключ ответов

№ вопроса в тесте	Номер ответа (или ответ, или соответствие)
1	b, d
2	b
3	a
4	a, b

№ вопроса в тесте	Номер ответа (или ответ, или соответствие)
5	a
6	a
7	a, b
8	a
9	a, d
10	c
11	b
12	b
13	a
14	b, d
15	b
16	c, d
17	b
18	a
19	d
20	a, c
21	a, b, c, d
22	c
23	b, d
24	c
25	b
26	c
27	b
28	b
29	b

Перечень тестовых вопросов

№ 1. Задание с множественным выбором. Выберите 2 правильных ответа.

Что такое автономный агент?

a. Сущность, действия которой основываются только на данных изначально заложенных при создании.

b. это интеллектуальный агент, который принимает свои решения о необходимых действиях без влияния из вне.

c. Сущность, действия которой направлены на достижение поставленной цели, не взаимодействуя с окружением.

d. Сущность, действия которой направлены на достижение поставленной цели, взаимодействуя с окружением используя сенсоры и актюаторы.

№ 2. Задание с единственным выбором. Выберите один правильный ответ.

Для чего используется ключевое слово puppet?

a. для обозначение дочернего класса

- b. для синхронизации состояний
- c. для создания удаленного клиента

№ 3. Задание с единственным выбором. Выберите один правильный ответ.

Для чего применяются аннотации в языке GDScript?

a. аннотации необходимы для внешних утилит и определяют то, как будет обработан скрипт

- b. аннотации используются для документирования кода
- c. аннотации меняют поведение метода или функции аналогично декоратору

№ 4. Задание с множественным выбором. Выберите 2 правильных ответа.

Какие игры можно разрабатывать на движке Godot?

- a. 3D
- b. 2D

№ 5. Задание с единственным выбором. Выберите один правильный ответ.

Что такое RPC?

- a. удаленный вызов процедур
- b. удаленное соединение игрока

№ 6. Задание с единственным выбором. Выберите один правильный ответ.

Какой метод используется для рисования объекта?

- a. `_draw`
- b. `_set_texture`
- c. `_process`

№ 7. Задание с множественным выбором. Выберите 2 правильных ответа.

Что такое компьютерная игра?

- a. Игра, построенная с использованием мультимедийных возможностей компьютера.
- b. Сложная программа, которая служит человеку для обучения или развлечения и рассчитана на разные возрастные группы.

№ 8. Задание с единственным выбором. Выберите один правильный ответ.

Верно ли утверждение, что язык GDScript является интерпретируемым языком программирования?

- a. Да
- b. Нет

№ 9. Задание с множественным выбором. Выберите 2 правильных ответа.

Какими свойствами можно настраивать генерацию шума на основе волн?

- a. частота
- b. сдвиг
- c. скорость
- d. амплитуда

№ 10. Задание с единственным выбором. Выберите один правильный ответ.

Каким образом изменится поведение частиц при установки параметра Local coords в disabled?

- a. Частицы будут испускаться из центра области и не будут выходить за нее

b. Частицы перестанут испускаться

c. Частицы будут испускаться из центра области и будут выходить за нее

№ 11. Задание с единичным выбором. Выберите один правильный ответ.

Каким образом можно сделать так, чтобы анимация не зависела от FPS?

a. ничего не нужно делать, так как анимация не зависит от количества выводимых кадров

b. необходимо учесть время между кадрами и рассчитать относительно него время между кадрами анимации

№ 12. Задание с единичным выбором. Выберите один правильный ответ.

Верно ли утверждение, что любой объект в Godot представлен в виде объекта типа Scene?

a. Нет

b. Да

№ 13. Задание с единичным выбором. Выберите один правильный ответ.

Какой параметр можно использовать для того, чтобы частицы вращались вокруг своей оси после испускания?

a. Angular velocity

b. Orbit velocity

c. Spin velocity

№ 14. Задание с множественным выбором. Выберите 2 правильных ответа.

Какое смещение в координатах необходимо учитывать при рисовании линий или прямоугольников с нечетной толщиной?

a. -1.

b. 0.5

c. 1.

d. -0.5

№ 15. Задание с единичным выбором. Выберите один правильный ответ.

Какой используется механизм для обработки возникающих событий?

a. слотов

b. сигналов

c. машина состояний

№ 16. Задание с множественным выбором. Выберите 2 правильных ответа.

Какое смещение в координатах необходимо учитывать при рисовании линий или прямоугольников с нечетной толщиной?

a. -1.

b. 1.

c. -0.5

d. 0.5

№ 17. Задание с единичным выбором. Выберите один правильный ответ.

Можно ли установить несколько обработчиков для одного сигнала?

a. Нет

b. Да

№ 18. Задание с единичным выбором. Выберите один правильный ответ.

Так как метод `_draw` кэшируется, то для рисования в зависимости от состояния необходимо использовать дополнительный метод. Какой?

- a. `queue_redraw`
- b. `paint`
- c. `draw`
- d. `_redraw`

№ 19. Задание с единичным выбором. Выберите один правильный ответ.

Какой ключевой объект используется для построения игр?

- a. `Context`
- b. `Node`
- c. `Engine`
- d. `Scene`

№ 20. Задание с множественным выбором. Выберите 2 правильных ответа.

Какие скриптовые языки можно использовать в Godot?

- a. GDScript
- b. Python
- c. C#

№ 21. Задание с множественным выбором. Выберите 4 правильных ответа.

Какие основные слои используются в автономных агентах?

- a. Слой действия
- b. Слой управления
- c. Слой движения
- d. Слой выбора

№ 22. Задание с единичным выбором. Выберите один правильный ответ.

Какой параметр отвечает за время жизни частицы?

- a. `decay`
- b. `health`
- c. `lifetime`
- d. `life`

№ 23. Задание с множественным выбором. Выберите 2 правильных ответа.

Какие форматы можно использовать чтобы загрузить трехмерную сцену в Godot?

- a. `.cad`
- b. `.obj`
- c. `.3ds`
- d. `.blend`

№ 24. Задание с единичным выбором. Выберите один правильный ответ.

В каких величинах измеряется громкость звука?

- a. См
- b. Гц

c. дБ

d. Кг

№ 25. Задание с единичным выбором. Выберите один правильный ответ.

Какой используется механизм для обработки возникающих событий?

a. слотов

b. сигналов

c. машина состояний

№ 26. Задание с единичным выбором. Выберите один правильный ответ.

Что такое keyframe?

a. начальный кадр анимации

b. конечный кадр анимации

c. каждый кадр в анимации остающийся неизменным

№ 27. Задание с единичным выбором. Выберите один правильный ответ.

Какой объект используется для создания звука идущего из определенной точки?

a. AudioBus

b. AudioStream

c. AudioChannel

d. AudioSpot

№ 28. Задание с единичным выбором. Выберите один правильный ответ.

На каком типе данных основаны сигналы?

a. InputEvent

b. Variant

c. Scene

d. Quaternion

№ 29. Задание с единичным выбором. Выберите один правильный ответ.

Может ли сигнал принимать параметры?

a. Нет

b. Да

2.3.2. Практические задания для оценки компетенции «ПК-1 ППК-Р1.1»

№ 1. Разработать алгоритм движения игрового объекта по случайному закону.

№ 2. Основные конструкции языка и их применение.

№ 3. Разработать игру Ping-Pong.

№ 4. Написать игру "Flappy Bird".

№ 5. При помощи системы частиц создать эффект огня.

№ 6. Кейс: Разработка 2D-аркады «Перехват солнечного ветра» для популяризации гелиофизики.

Научный институт солнечно-земной физики готовит выставку для школьников и абитуриентов и заказывает небольшую 2D-игру для интерактивного киоска. Игра должна в увлекательной форме показать, как работает система мониторинга космической погоды: игрок управляет орбитальным спутником-детектором, который перехватывает частицы солнечного ветра и классифицирует их по типам. Студенты выступают в роли начинающих

разработчиков игр.

На выставке посетителям сложно объяснить, чем занимаются учёные-гелиофизики, с помощью одних лишь плакатов. Нужна простая, но наглядная игра, которая за 2–3 минуты объяснит базовые понятия: солнечный ветер, типы частиц (протоны, электроны, альфа-частицы), роль спутников-мониторов. Игра должна быть лёгкой в освоении, работать на обычном ПК и не требовать сложной графики.

Разработать прототип 2D-аркады на любом игровом движке Godot Реализовать: управление спутником с клавиатуры или мыши, спавн частиц солнечного ветра с разными свойствами, систему классификации частиц по типам (захват нужным детектором), подсчёт очков и уровней сложности, простое меню и экран окончания игры.

Краткая справка о солнечном ветре и типах частиц (для игрового контекста); набор простых 2D-спрайтов (спутник, частицы трёх типов, фон звёздного неба); геймдизайн-документ с описанием механик и баланса; примеры похожих простых 2D-игр (Arkanoid, Space Invaders) как референс; базовые уроки по работе с 2D в выбранном движке.

Работающий прототип 2D-аркады, который можно запустить на обычном ПК; исходный код проекта с комментариями; краткое описание реализованных механик и логики игры; короткая презентация с демонстрацией геймплея и объяснением, как игра решает задачу популяризации науки.

Соответствие решению поставленной задаче — 40% (все заявленные механики работают, игра понятна); качество реализации и чистота кода — 25% (архитектура, отсутствие критических багов, читаемость); игровой баланс и UX — 20% (удобство управления, понятность правил, плавность геймплея); качество презентации и защиты — 15% (ясность объяснения решений и демонстрация).

№ 7. Разработать машину состояний и на ее основе реализовать анимацию игрового персонажа.

№ 8. Написать клиента для игры "выживание".

№ 9. Кейс: Мультиплеерная кооперативная игра для совместного управления сетью радаров.

Институт располагает распределённой сетью из нескольких когерентных радаров, разнесённых географически. Для полноценного мониторинга ионосферы операторы радаров должны координировать свои действия в реальном времени. Институт заказывает разработку кооперативной мультиплеерной игры-тренажёра, где команда из 3–5 игроков совместно управляет виртуальной радарной сетью, реагируя на геомагнитные возмущения. Студенты — команда сетевых разработчиков.

Реальные операторы радаров работают удалённо и нуждаются в отработке навыков командного взаимодействия в стрессовых ситуациях (геомагнитные бури). Существующие тренажёры рассчитаны на одного пользователя и не моделируют сетевое взаимодействие. Разработка мультиплеера сопряжена с проблемами синхронизации состояния игры, задержек сети (latency), разрешения конфликтов при одновременных действиях игроков и обеспечения плавного игрового опыта.

Разработать прототип кооперативной мультиплеерной игры с сетевой архитектурой (клиент-сервер или peer-to-peer с авторитетным сервером). Реализовать: синхронизацию состояния радаров между игроками, систему ролей (каждый игрок управляет своим радаром), механику совместного покрытия зон наблюдения, механизм компенсации сетевых задержек (lag compensation / client-side prediction), систему голосовой или текстовой коммуникации между игроками, сценарии геомагнитных бурь с нарастающей сложностью.

Карта размещения реальных радаров с параметрами (дальность, угол обзора, частота); сценарии геомагнитных возмущений (таймлайны событий); библиотека сетевых фреймворков для выбранного движка (Mirror, Photon, Netcode for GameObjects и др.); требования к допустимой задержке (<100 мс); тестовые сценарии кооперативного геймплея; описание UX-паттернов для кооперативных интерфейсов.

Работающий сетевой прототип с подключением 3–5 игроков; демонстрация синхронизированного управления радаром в реальном времени; визуализация зон покрытия и их объединения; стресс-тест с искусственно добавленной задержкой и отчёт о стабильности; документация по сетевой архитектуре и выбору модели синхронизации; презентация.

Сетевая архитектура и синхронизация — 35% (стабильность, корректность, устойчивость к задержкам); игровой дизайн и кооперативные механики — 25% (вовлечённость, баланс, необходимость командного взаимодействия); качество кода и масштабируемость — 25% (чистота архитектуры, возможность добавления новых радаров и сценариев); качество презентации и защиты — 15%.

№ 10. Разработать алгоритм окружения врагами игрока.

№ 11. Кейс: Процедурная генерация космических явлений и ИИ-агенты для образовательной стратегии по солнечно-земной физике.

Институт солнечно-земной физики разрабатывает образовательную компьютерную игру в жанре стратегии, в которой игрок управляет исследовательским центром, изучающим влияние солнечной активности на Землю. В игре необходимо моделировать солнечные вспышки, корональные выбросы массы и их воздействие на магнитосферу, ионосферу и инфраструктуру Земли. Студенты выступают в роли разработчиков игровых систем.

Солнечная активность непредсказуема и многофакторна: каждая вспышка уникальна по интенсивности, направлению и составу. Предзаданные сценарии быстро надоедают игроку и не отражают реальную стохастическую природу явлений. Также необходимо, чтобы виртуальные «научные сотрудники» (ИИ-агенты) реалистично реагировали на события, принимали решения и взаимодействовали с игроком. Требуются система процедурной генерации и интеллектуальное поведение NPC.

Разработать систему процедурной генерации солнечных событий (вспышки классов А–Х, корональные выбросы, потоки солнечного ветра) с параметрами, основанными на реальных статистических распределениях. Реализовать ИИ-агентов (научных сотрудников обсерватории), использующих деревья поведения (Behavior Trees) или конечные автоматы, которые анализируют сгенерированные события, выдают рекомендации игроку и реагируют на его команды. Интегрировать обе системы в игровой прототип.

Датасет реальных солнечных событий (GOES, NOAA) для калибровки генератора; описание физических параметров явлений; шаблоны деревьев поведения и FSM; игровой прототип с

базовой архитектурой; геймдизайн-документ с описанием ролей NPC и игрового цикла; API игрового движка для работы с procedural generation и AI.

Модуль процедурной генерации с настраиваемыми параметрами распределений; система ИИ-агентов с визуализацией логики принятия решений; интегрированный прототип с демонстрацией взаимодействия агентов и генерируемых событий; отчет с описанием алгоритмов и обоснованием выбора архитектуры ИИ; презентация.

Качество процедурной генерации — 30% (разнообразие, реалистичность, параметризуемость); архитектура и поведение ИИ-агентов — 30% (логичность, адаптивность, читаемость кода); интеграция и работоспособность прототипа — 25% (отсутствие критических багов, согласованность систем); качество презентации и защиты — 15%.

№ 12. Написать генератор лабиринта.

2.3.3. Практические задания для оценки компетенции «ПК-1 ППК-Р1.2»

№ 13. Разработать алгоритм движения игрового объекта по случайному закону.

№ 14. Основные конструкции языка и их применение.

№ 15. Разработать игру Ping-Pong.

№ 16. Реализовать вращение объекта на заданный угол при помощи пользовательского ввода.

№ 17. Написать игру "Flappy Bird".

№ 18. При помощи системы частиц создать эффект огня.

№ 19. Кейс: Разработка 2D-аркады «Перехват солнечного ветра» для популяризации гелиофизики.

Научный институт солнечно-земной физики готовит выставку для школьников и абитуриентов и заказывает небольшую 2D-игру для интерактивного киоска. Игра должна в увлекательной форме показать, как работает система мониторинга космической погоды: игрок управляет орбитальным спутником-детектором, который перехватывает частицы солнечного ветра и классифицирует их по типам. Студенты выступают в роли начинающих разработчиков игр.

На выставке посетителям сложно объяснить, чем занимаются учёные-гелиофизики, с помощью одних лишь плакатов. Нужна простая, но наглядная игра, которая за 2–3 минуты объяснит базовые понятия: солнечный ветер, типы частиц (протоны, электроны, альфа-частицы), роль спутников-мониторов. Игра должна быть лёгкой в освоении, работать на обычном ПК и не требовать сложной графики.

Разработать прототип 2D-аркады на любом игровом движке Godot Реализовать: управление спутником с клавиатуры или мыши, спавн частиц солнечного ветра с разными свойствами, систему классификации частиц по типам (захват нужным детектором), подсчёт очков и уровней сложности, простое меню и экран окончания игры.

Краткая справка о солнечном ветре и типах частиц (для игрового контекста); набор простых 2D-спрайтов (спутник, частицы трёх типов, фон звёздного неба); геймдизайн-документ с описанием механик и баланса; примеры похожих простых 2D-игр (Arkanoid, Space Invaders) как референс; базовые уроки по работе с 2D в выбранном движке.

Работающий прототип 2D-аркады, который можно запустить на обычном ПК; исходный код проекта с комментариями; краткое описание реализованных механик и логики игры; короткая презентация с демонстрацией геймплея и объяснением, как игра решает задачу популяризации науки.

Соответствие решению поставленной задаче — 40% (все заявленные механики работают, игра понятна); качество реализации и чистота кода — 25% (архитектура, отсутствие критических багов, читаемость); игровой баланс и UX — 20% (удобство управления, понятность правил, плавность геймплея); качество презентации и защиты — 15% (ясность объяснения решений и демонстрация).

№ 20. Разработать машину состояний и на ее основе реализовать анимацию игрового персонажа.

№ 21. Написать клиента для игры "выживание".

№ 22. Кейс: Мультиплеерная кооперативная игра для совместного управления сетью радаров.

Институт располагает распределённой сетью из нескольких когерентных радаров, разнесённых географически. Для полноценного мониторинга ионосферы операторы радаров должны координировать свои действия в реальном времени. Институт заказывает разработку кооперативной мультиплеерной игры-тренажёра, где команда из 3–5 игроков совместно управляет виртуальной радарной сетью, реагируя на геомагнитные возмущения. Студенты — команда сетевых разработчиков.

Реальные операторы радаров работают удалённо и нуждаются в отработке навыков командного взаимодействия в стрессовых ситуациях (геомагнитные бури). Существующие тренажёры рассчитаны на одного пользователя и не моделируют сетевое взаимодействие. Разработка мультиплеера сопряжена с проблемами синхронизации состояния игры, задержек сети (latency), разрешения конфликтов при одновременных действиях игроков и обеспечения плавного игрового опыта.

Разработать прототип кооперативной мультиплеерной игры с сетевой архитектурой (клиент-сервер или peer-to-peer с авторитетным сервером). Реализовать: синхронизацию состояния радаров между игроками, систему ролей (каждый игрок управляет своим радаром), механику совместного покрытия зон наблюдения, механизм компенсации сетевых задержек (lag compensation / client-side prediction), систему голосовой или текстовой коммуникации между игроками, сценарии геомагнитных бурь с нарастающей сложностью.

Карта размещения реальных радаров с параметрами (дальность, угол обзора, частота); сценарии геомагнитных возмущений (таймлайны событий); библиотека сетевых фреймворков для выбранного движка (Mirror, Photon, Netcode for GameObjects и др.); требования к допустимой задержке (<100 мс); тестовые сценарии кооперативного геймплея; описание UX-паттернов для кооперативных интерфейсов.

Работающий сетевой прототип с подключением 3–5 игроков; демонстрация синхронизированного управления радаром в реальном времени; визуализация зон покрытия и их объединения; стресс-тест с искусственно добавленной задержкой и отчёт о стабильности; документация по сетевой архитектуре и выбору модели синхронизации; презентация.

Сетевая архитектура и синхронизация — 35% (стабильность, корректность, устойчивость к задержкам); игровой дизайн и кооперативные механики — 25% (вовлечённость, баланс, необходимость командного взаимодействия); качество кода и масштабируемость — 25% (чистота архитектуры, возможность добавления новых радаров и сценариев); качество презентации и защиты — 15%.

№ 23. Разработать алгоритм окружения врагами игрока.

№ 24. Кейс: Процедурная генерация космических явлений и ИИ-агенты для образовательной стратегии по солнечно-земной физике.

Институт солнечно-земной физики разрабатывает образовательную компьютерную игру в жанре стратегии, в которой игрок управляет исследовательским центром, изучающим влияние солнечной активности на Землю. В игре необходимо моделировать солнечные вспышки, корональные выбросы массы и их воздействие на магнитосферу, ионосферу и инфраструктуру Земли. Студенты выступают в роли разработчиков игровых систем.

Солнечная активность непредсказуема и многофакторна: каждая вспышка уникальна по интенсивности, направлению и составу. Предзаданные сценарии быстро надоедают игроку и не отражают реальную стохастическую природу явлений. Также необходимо, чтобы виртуальные «научные сотрудники» (ИИ-агенты) реалистично реагировали на события, принимали решения и взаимодействовали с игроком. Требуется система процедурной генерации и интеллектуальное поведение NPC.

Разработать систему процедурной генерации солнечных событий (вспышки классов А–Х, корональные выбросы, потоки солнечного ветра) с параметрами, основанными на реальных статистических распределениях. Реализовать ИИ-агентов (научных сотрудников обсерватории), использующих деревья поведения (Behavior Trees) или конечные автоматы, которые анализируют сгенерированные события, выдают рекомендации игроку и реагируют на его команды. Интегрировать обе системы в игровой прототип.

Датасет реальных солнечных событий (GOES, NOAA) для калибровки генератора; описание физических параметров явлений; шаблоны деревьев поведения и FSM; игровой прототип с базовой архитектурой; геймдизайн-документ с описанием ролей NPC и игрового цикла; API игрового движка для работы с procedural generation и AI.

Модуль процедурной генерации с настраиваемыми параметрами распределений; система ИИ-агентов с визуализацией логики принятия решений; интегрированный прототип с демонстрацией взаимодействия агентов и генерируемых событий; отчет с описанием алгоритмов и обоснованием выбора архитектуры ИИ; презентация.

Качество процедурной генерации — 30% (разнообразие, реалистичность, параметризуемость); архитектура и поведение ИИ-агентов — 30% (логичность, адаптивность, читаемость кода); интеграция и работоспособность прототипа — 25% (отсутствие критических багов, согласованность систем); качество презентации и защиты — 15%.

№ 25. Написать генератор лабиринта.

2.3.4. Практические задания для оценки компетенции «ПК-1 ППК-Р1.3»

№ 26. Разработать алгоритм движения игрового объекта по случайному закону.

№ 27. Основные конструкции языка и их применение.

№ 28. Разработать игру *Ping-Pong*.

№ 29. Написать игру *"Flappy Bird"*.

№ 30. При помощи системы частиц создать эффект огня.

№ 31. Кейс: Разработка 2D-аркады «Перехват солнечного ветра» для популяризации гелиофизики.

Научный институт солнечно-земной физики готовит выставку для школьников и абитуриентов и заказывает небольшую 2D-игру для интерактивного киоска. Игра должна в увлекательной форме показать, как работает система мониторинга космической погоды: игрок управляет орбитальным спутником-детектором, который перехватывает частицы солнечного ветра и классифицирует их по типам. Студенты выступают в роли начинающих разработчиков игр.

На выставке посетителям сложно объяснить, чем занимаются учёные-гелиофизики, с помощью одних лишь плакатов. Нужна простая, но наглядная игра, которая за 2–3 минуты объяснит базовые понятия: солнечный ветер, типы частиц (протоны, электроны, альфа-частицы), роль спутников-мониторов. Игра должна быть лёгкой в освоении, работать на обычном ПК и не требовать сложной графики.

Разработать прототип 2D-аркады на любом игровом движке Godot Реализовать: управление спутником с клавиатуры или мыши, спавн частиц солнечного ветра с разными свойствами, систему классификации частиц по типам (захват нужным детектором), подсчёт очков и уровней сложности, простое меню и экран окончания игры.

Краткая справка о солнечном ветре и типах частиц (для игрового контекста); набор простых 2D-спрайтов (спутник, частицы трёх типов, фон звёздного неба); геймдизайн-документ с описанием механик и баланса; примеры похожих простых 2D-игр (Arkanoid, Space Invaders) как референс; базовые уроки по работе с 2D в выбранном движке.

Работающий прототип 2D-аркады, который можно запустить на обычном ПК; исходный код проекта с комментариями; краткое описание реализованных механик и логики игры; короткая презентация с демонстрацией геймплея и объяснением, как игра решает задачу популяризации науки.

Соответствие решению поставленной задаче — 40% (все заявленные механики работают, игра понятна); качество реализации и чистота кода — 25% (архитектура, отсутствие критических багов, читаемость); игровой баланс и UX — 20% (удобство управления, понятность правил, плавность геймплея); качество презентации и защиты — 15% (ясность объяснения решений и демонстрация).

№ 32. Разработать машину состояний и на ее основе реализовать анимацию игрового персонажа.

№ 33. Написать клиента для игры *"выживание"*.

№ 34. Кейс: Мультиплеерная кооперативная игра для совместного управления сетью радаров.

Институт располагает распределённой сетью из нескольких когерентных радаров, разнесённых географически. Для полноценного мониторинга ионосферы операторы радаров должны координировать свои действия в реальном времени. Институт заказывает разработку кооперативной мультиплеерной игры-тренажёра, где команда из 3–5 игроков совместно

управляет виртуальной радарной сетью, реагируя на геомагнитные возмущения. Студенты — команда сетевых разработчиков.

Реальные операторы радаров работают удалённо и нуждаются в отработке навыков командного взаимодействия в стрессовых ситуациях (геомагнитные бури). Существующие тренажёры рассчитаны на одного пользователя и не моделируют сетевое взаимодействие. Разработка мультиплеера сопряжена с проблемами синхронизации состояния игры, задержек сети (latency), разрешения конфликтов при одновременных действиях игроков и обеспечения плавного игрового опыта.

Разработать прототип кооперативной мультиплеерной игры с сетевой архитектурой (клиент-сервер или peer-to-peer с авторитетным сервером). Реализовать: синхронизацию состояния радаров между игроками, систему ролей (каждый игрок управляет своим радаром), механику совместного покрытия зон наблюдения, механизм компенсации сетевых задержек (lag compensation / client-side prediction), систему голосовой или текстовой коммуникации между игроками, сценарии геомагнитных бурь с нарастающей сложностью.

Карта размещения реальных радаров с параметрами (дальность, угол обзора, частота); сценарии геомагнитных возмущений (таймлайны событий); библиотека сетевых фреймворков для выбранного движка (Mirror, Photon, Netcode for GameObjects и др.); требования к допустимой задержке (<100 мс); тестовые сценарии кооперативного геймплея; описание UX-паттернов для кооперативных интерфейсов.

Работающий сетевой прототип с подключением 3–5 игроков; демонстрация синхронизированного управления радарами в реальном времени; визуализация зон покрытия и их объединения; стресс-тест с искусственно добавленной задержкой и отчёт о стабильности; документация по сетевой архитектуре и выбору модели синхронизации; презентация.

Сетевая архитектура и синхронизация — 35% (стабильность, корректность, устойчивость к задержкам); игровой дизайн и кооперативные механики — 25% (вовлечённость, баланс, необходимость командного взаимодействия); качество кода и масштабируемость — 25% (чистота архитектуры, возможность добавления новых радаров и сценариев); качество презентации и защиты — 15%.

№ 35. Разработать алгоритм окружения врагами игрока.

№ 36. Кейс: Процедурная генерация космических явлений и ИИ-агенты для образовательной стратегии по солнечно-земной физике.

Институт солнечно-земной физики разрабатывает образовательную компьютерную игру в жанре стратегии, в которой игрок управляет исследовательским центром, изучающим влияние солнечной активности на Землю. В игре необходимо моделировать солнечные вспышки, корональные выбросы массы и их воздействие на магнитосферу, ионосферу и инфраструктуру Земли. Студенты выступают в роли разработчиков игровых систем.

Солнечная активность непредсказуема и многофакторна: каждая вспышка уникальна по интенсивности, направлению и составу. Предзаданные сценарии быстро надоедают игроку и не отражают реальную стохастическую природу явлений. Также необходимо, чтобы виртуальные «научные сотрудники» (ИИ-агенты) реалистично реагировали на события, принимали решения и взаимодействовали с игроком. Требуется система процедурной генерации и интеллектуальное поведение NPC.

Разработать систему процедурной генерации солнечных событий (вспышки классов А–Х, корональные выбросы, потоки солнечного ветра) с параметрами, основанными на реальных статистических распределениях. Реализовать ИИ-агентов (научных сотрудников обсерватории), использующих деревья поведения (Behavior Trees) или конечные автоматы, которые анализируют сгенерированные события, выдают рекомендации игроку и реагируют на его команды. Интегрировать обе системы в игровой прототип.

Датасет реальных солнечных событий (GOES, NOAA) для калибровки генератора; описание физических параметров явлений; шаблоны деревьев поведения и FSM; игровой прототип с базовой архитектурой; геймдизайн-документ с описанием ролей NPC и игрового цикла; API игрового движка для работы с procedural generation и AI.

Модуль процедурной генерации с настраиваемыми параметрами распределений; система ИИ-агентов с визуализацией логики принятия решений; интегрированный прототип с демонстрацией взаимодействия агентов и генерируемых событий; отчет с описанием алгоритмов и обоснованием выбора архитектуры ИИ; презентация.

Качество процедурной генерации — 30% (разнообразие, реалистичность, параметризуемость); архитектура и поведение ИИ-агентов — 30% (логичность, адаптивность, читаемость кода); интеграция и работоспособность прототипа — 25% (отсутствие критических багов, согласованность систем); качество презентации и защиты — 15%.

№ 37. Написать генератор лабиринта.

2.3.5. Практические задания для оценки компетенции «ПК-2 ППК-Р2.1»

№ 38. Разработать алгоритм движения игрового объекта по случайному закону.

№ 39. Основные конструкции языка и их применение.

№ 40. Разработать игру Ping-Pong.

№ 41. Реализовать вращение объекта на заданный угол при помощи пользовательского ввода.

№ 42. Написать игру "Flappy Bird".

№ 43. При помощи системы частиц создать эффект огня.

№ 44. Кейс: Разработка 2D-аркады «Перехват солнечного ветра» для популяризации гелиофизики.

Научный институт солнечно-земной физики готовит выставку для школьников и абитуриентов и заказывает небольшую 2D-игру для интерактивного киоска. Игра должна в увлекательной форме показать, как работает система мониторинга космической погоды: игрок управляет орбитальным спутником-детектором, который перехватывает частицы солнечного ветра и классифицирует их по типам. Студенты выступают в роли начинающих разработчиков игр.

На выставке посетителям сложно объяснить, чем занимаются учёные-гелиофизики, с помощью одних лишь плакатов. Нужна простая, но наглядная игра, которая за 2–3 минуты объяснит базовые понятия: солнечный ветер, типы частиц (протоны, электроны, альфа-частицы), роль спутников-мониторов. Игра должна быть лёгкой в освоении, работать на обычном ПК и не требовать сложной графики.

Разработать прототип 2D-аркады на любом игровом движке Godot Реализовать: управление

спутником с клавиатуры или мыши, спавн частиц солнечного ветра с разными свойствами, систему классификации частиц по типам (захват нужным детектором), подсчёт очков и уровней сложности, простое меню и экран окончания игры.

Краткая справка о солнечном ветре и типах частиц (для игрового контекста); набор простых 2D-спрайтов (спутник, частицы трёх типов, фон звёздного неба); геймдизайн-документ с описанием механик и баланса; примеры похожих простых 2D-игр (Arkanoid, Space Invaders) как референс; базовые уроки по работе с 2D в выбранном движке.

Работающий прототип 2D-аркады, который можно запустить на обычном ПК; исходный код проекта с комментариями; краткое описание реализованных механик и логики игры; короткая презентация с демонстрацией геймплея и объяснением, как игра решает задачу популяризации науки.

Соответствие решению поставленной задаче — 40% (все заявленные механики работают, игра понятна); качество реализации и чистота кода — 25% (архитектура, отсутствие критических багов, читаемость); игровой баланс и UX — 20% (удобство управления, понятность правил, плавность геймплея); качество презентации и защиты — 15% (ясность объяснения решений и демонстрация).

№ 45. Разработать машину состояний и на ее основе реализовать анимацию игрового персонажа.

№ 46. Написать клиента для игры "выживание".

№ 47. Кейс: Мультиплеерная кооперативная игра для совместного управления сетью радаров.

Институт располагает распределённой сетью из нескольких когерентных радаров, разнесённых географически. Для полноценного мониторинга ионосферы операторы радаров должны координировать свои действия в реальном времени. Институт заказывает разработку кооперативной мультиплеерной игры-тренажёра, где команда из 3–5 игроков совместно управляет виртуальной радарной сетью, реагируя на геомагнитные возмущения. Студенты — команда сетевых разработчиков.

Реальные операторы радаров работают удалённо и нуждаются в отработке навыков командного взаимодействия в стрессовых ситуациях (геомагнитные бури). Существующие тренажёры рассчитаны на одного пользователя и не моделируют сетевое взаимодействие. Разработка мультиплеера сопряжена с проблемами синхронизации состояния игры, задержек сети (latency), разрешения конфликтов при одновременных действиях игроков и обеспечения плавного игрового опыта.

Разработать прототип кооперативной мультиплеерной игры с сетевой архитектурой (клиент-сервер или peer-to-peer с авторитетным сервером). Реализовать: синхронизацию состояния радаров между игроками, систему ролей (каждый игрок управляет своим радаром), механику совместного покрытия зон наблюдения, механизм компенсации сетевых задержек (lag compensation / client-side prediction), систему голосовой или текстовой коммуникации между игроками, сценарии геомагнитных бурь с нарастающей сложностью.

Карта размещения реальных радаров с параметрами (дальность, угол обзора, частота); сценарии геомагнитных возмущений (таймлайны событий); библиотека сетевых

фреймворков для выбранного движка (Mirror, Photon, Netcode for GameObjects и др.); требования к допустимой задержке (<100 мс); тестовые сценарии кооперативного геймплея; описание UX-паттернов для кооперативных интерфейсов.

Работающий сетевой прототип с подключением 3–5 игроков; демонстрация синхронизированного управления радаром в реальном времени; визуализация зон покрытия и их объединения; стресс-тест с искусственно добавленной задержкой и отчет о стабильности; документация по сетевой архитектуре и выбору модели синхронизации; презентация.

Сетевая архитектура и синхронизация — 35% (стабильность, корректность, устойчивость к задержкам); игровой дизайн и кооперативные механики — 25% (вовлечённость, баланс, необходимость командного взаимодействия); качество кода и масштабируемость — 25% (чистота архитектуры, возможность добавления новых радаров и сценариев); качество презентации и защиты — 15%.

№ 48. Разработать алгоритм окружения врагами игрока.

№ 49. Кейс: Процедурная генерация космических явлений и ИИ-агенты для образовательной стратегии по солнечно-земной физике.

Институт солнечно-земной физики разрабатывает образовательную компьютерную игру в жанре стратегии, в которой игрок управляет исследовательским центром, изучающим влияние солнечной активности на Землю. В игре необходимо моделировать солнечные вспышки, корональные выбросы массы и их воздействие на магнитосферу, ионосферу и инфраструктуру Земли. Студенты выступают в роли разработчиков игровых систем.

Солнечная активность непредсказуема и многофакторна: каждая вспышка уникальна по интенсивности, направлению и составу. Предзаданные сценарии быстро надоедают игроку и не отражают реальную стохастическую природу явлений. Также необходимо, чтобы виртуальные «научные сотрудники» (ИИ-агенты) реалистично реагировали на события, принимали решения и взаимодействовали с игроком. Требуется система процедурной генерации и интеллектуальное поведение NPC.

Разработать систему процедурной генерации солнечных событий (вспышки классов А–Х, корональные выбросы, потоки солнечного ветра) с параметрами, основанными на реальных статистических распределениях. Реализовать ИИ-агентов (научных сотрудников обсерватории), использующих деревья поведения (Behavior Trees) или конечные автоматы, которые анализируют сгенерированные события, выдают рекомендации игроку и реагируют на его команды. Интегрировать обе системы в игровой прототип.

Датасет реальных солнечных событий (GOES, NOAA) для калибровки генератора; описание физических параметров явлений; шаблоны деревьев поведения и FSM; игровой прототип с базовой архитектурой; геймдизайн-документ с описанием ролей NPC и игрового цикла; API игрового движка для работы с procedural generation и AI.

Модуль процедурной генерации с настраиваемыми параметрами распределений; система ИИ-агентов с визуализацией логики принятия решений; интегрированный прототип с демонстрацией взаимодействия агентов и генерируемых событий; отчет с описанием алгоритмов и обоснованием выбора архитектуры ИИ; презентация.

Качество процедурной генерации — 30% (разнообразие, реалистичность, параметризуе-

мость); архитектура и поведение ИИ-агентов — 30% (логичность, адаптивность, читаемость кода); интеграция и работоспособность прототипа — 25% (отсутствие критических багов, согласованность систем); качество презентации и защиты — 15%.

№ 50. Написать генератор лабиринта.

2.3.6. Практические задания для оценки компетенции «ПК-2 ППК-Р2.2»

№ 51. Разработать алгоритм движения игрового объекта по случайному закону.

№ 52. Основные конструкции языка и их применение.

№ 53. Разработать игру Ping-Pong.

№ 54. Реализовать вращение объекта на заданный угол при помощи пользовательского ввода.

№ 55. Написать игру "Flappy Bird".

№ 56. При помощи системы частиц создать эффект огня.

№ 57. Кейс: Разработка 2D-аркады «Перехват солнечного ветра» для популяризации гелиофизики.

Научный институт солнечно-земной физики готовит выставку для школьников и абитуриентов и заказывает небольшую 2D-игру для интерактивного киоска. Игра должна в увлекательной форме показать, как работает система мониторинга космической погоды: игрок управляет орбитальным спутником-детектором, который перехватывает частицы солнечного ветра и классифицирует их по типам. Студенты выступают в роли начинающих разработчиков игр.

На выставке посетителям сложно объяснить, чем занимаются учёные-гелиофизики, с помощью одних лишь плакатов. Нужна простая, но наглядная игра, которая за 2–3 минуты объяснит базовые понятия: солнечный ветер, типы частиц (протоны, электроны, альфа-частицы), роль спутников-мониторов. Игра должна быть лёгкой в освоении, работать на обычном ПК и не требовать сложной графики.

Разработать прототип 2D-аркады на любом игровом движке Godot Реализовать: управление спутником с клавиатуры или мыши, спавн частиц солнечного ветра с разными свойствами, систему классификации частиц по типам (захват нужным детектором), подсчёт очков и уровней сложности, простое меню и экран окончания игры.

Краткая справка о солнечном ветре и типах частиц (для игрового контекста); набор простых 2D-спрайтов (спутник, частицы трёх типов, фон звёздного неба); геймдизайн-документ с описанием механик и баланса; примеры похожих простых 2D-игр (Arkanoid, Space Invaders) как референс; базовые уроки по работе с 2D в выбранном движке.

Работающий прототип 2D-аркады, который можно запустить на обычном ПК; исходный код проекта с комментариями; краткое описание реализованных механик и логики игры; короткая презентация с демонстрацией геймплея и объяснением, как игра решает задачу популяризации науки.

Соответствие решению поставленной задаче — 40% (все заявленные механики работают, игра понятна); качество реализации и чистота кода — 25% (архитектура, отсутствие критических багов, читаемость); игровой баланс и UX — 20% (удобство управления, понятность правил, плавность геймплея); качество презентации и защиты — 15% (ясность объяснения

решений и демонстрация).

№ 58. Разработать машину состояний и на ее основе реализовать анимацию игрового персонажа.

№ 59. Написать клиента для игры "выживание".

№ 60. Кейс: Мультиплеерная кооперативная игра для совместного управления сетью радаров.

Институт располагает распределённой сетью из нескольких когерентных радаров, разнесённых географически. Для полноценного мониторинга ионосферы операторы радаров должны координировать свои действия в реальном времени. Институт заказывает разработку кооперативной мультиплеерной игры-тренажёра, где команда из 3–5 игроков совместно управляет виртуальной радарной сетью, реагируя на геомагнитные возмущения. Студенты — команда сетевых разработчиков.

Реальные операторы радаров работают удалённо и нуждаются в отработке навыков командного взаимодействия в стрессовых ситуациях (геомагнитные бури). Существующие тренажёры рассчитаны на одного пользователя и не моделируют сетевое взаимодействие. Разработка мультиплеера сопряжена с проблемами синхронизации состояния игры, задержек сети (latency), разрешения конфликтов при одновременных действиях игроков и обеспечения плавного игрового опыта.

Разработать прототип кооперативной мультиплеерной игры с сетевой архитектурой (клиент-сервер или peer-to-peer с авторитетным сервером). Реализовать: синхронизацию состояния радаров между игроками, систему ролей (каждый игрок управляет своим радаром), механику совместного покрытия зон наблюдения, механизм компенсации сетевых задержек (lag compensation / client-side prediction), систему голосовой или текстовой коммуникации между игроками, сценарии геомагнитных бурь с нарастающей сложностью.

Карта размещения реальных радаров с параметрами (дальность, угол обзора, частота); сценарии геомагнитных возмущений (таймлайны событий); библиотека сетевых фреймворков для выбранного движка (Mirror, Photon, Netcode for GameObjects и др.); требования к допустимой задержке (<100 мс); тестовые сценарии кооперативного геймплея; описание UX-паттернов для кооперативных интерфейсов.

Работающий сетевой прототип с подключением 3–5 игроков; демонстрация синхронизированного управления радаром в реальном времени; визуализация зон покрытия и их объединения; стресс-тест с искусственно добавленной задержкой и отчёт о стабильности; документация по сетевой архитектуре и выбору модели синхронизации; презентация.

Сетевая архитектура и синхронизация — 35% (стабильность, корректность, устойчивость к задержкам); игровой дизайн и кооперативные механики — 25% (вовлечённость, баланс, необходимость командного взаимодействия); качество кода и масштабируемость — 25% (чистота архитектуры, возможность добавления новых радаров и сценариев); качество презентации и защиты — 15%.

№ 61. Разработать алгоритм окружения врагами игрока.

№ 62. Кейс: Процедурная генерация космических явлений и ИИ-агенты для образовательной стратегии по солнечно-земной физике.

Институт солнечно-земной физики разрабатывает образовательную компьютерную

игру в жанре стратегии, в которой игрок управляет исследовательским центром, изучающим влияние солнечной активности на Землю. В игре необходимо моделировать солнечные вспышки, корональные выбросы массы и их воздействие на магнитосферу, ионосферу и инфраструктуру Земли. Студенты выступают в роли разработчиков игровых систем.

Солнечная активность непредсказуема и многофакторна: каждая вспышка уникальна по интенсивности, направлению и составу. Предзаданные сценарии быстро надоедают игроку и не отражают реальную стохастическую природу явлений. Также необходимо, чтобы виртуальные «научные сотрудники» (ИИ-агенты) реалистично реагировали на события, принимали решения и взаимодействовали с игроком. Требуются система процедурной генерации и интеллектуальное поведение NPC.

Разработать систему процедурной генерации солнечных событий (вспышки классов А–Х, корональные выбросы, потоки солнечного ветра) с параметрами, основанными на реальных статистических распределениях. Реализовать ИИ-агентов (научных сотрудников обсерватории), использующих деревья поведения (Behavior Trees) или конечные автоматы, которые анализируют сгенерированные события, выдают рекомендации игроку и реагируют на его команды. Интегрировать обе системы в игровой прототип.

Датасет реальных солнечных событий (GOES, NOAA) для калибровки генератора; описание физических параметров явлений; шаблоны деревьев поведения и FSM; игровой прототип с базовой архитектурой; геймдизайн-документ с описанием ролей NPC и игрового цикла; API игрового движка для работы с procedural generation и AI.

Модуль процедурной генерации с настраиваемыми параметрами распределений; система ИИ-агентов с визуализацией логики принятия решений; интегрированный прототип с демонстрацией взаимодействия агентов и генерируемых событий; отчет с описанием алгоритмов и обоснованием выбора архитектуры ИИ; презентация.

Качество процедурной генерации — 30% (разнообразие, реалистичность, параметризуемость); архитектура и поведение ИИ-агентов — 30% (логичность, адаптивность, читаемость кода); интеграция и работоспособность прототипа — 25% (отсутствие критических багов, согласованность систем); качество презентации и защиты — 15%.

№ 63. Написать генератор лабиринта.

2.3.7. Практические задания для оценки компетенции «ПК-2 ППК-Р2.3»

№ 64. Разработать алгоритм движения игрового объекта по случайному закону.

№ 65. Основные конструкции языка и их применение.

№ 66. Разработать игру Ping-Pong.

№ 67. Реализовать вращение объекта на заданный угол при помощи пользовательского ввода.

№ 68. Написать игру "Flappy Bird".

№ 69. При помощи системы частиц создать эффект огня.

№ 70. Кейс: Разработка 2D-аркады «Перехват солнечного ветра» для популяризации гелиофизики.

Научный институт солнечно-земной физики готовит выставку для школьников и абитуриентов и заказывает небольшую 2D-игру для интерактивного киоска. Игра должна в

увлекательной форме показать, как работает система мониторинга космической погоды: игрок управляет орбитальным спутником-детектором, который перехватывает частицы солнечного ветра и классифицирует их по типам. Студенты выступают в роли начинающих разработчиков игр.

На выставке посетителям сложно объяснить, чем занимаются учёные-гелиофизики, с помощью одних лишь плакатов. Нужна простая, но наглядная игра, которая за 2–3 минуты объяснит базовые понятия: солнечный ветер, типы частиц (протоны, электроны, альфа-частицы), роль спутников-мониторов. Игра должна быть лёгкой в освоении, работать на обычном ПК и не требовать сложной графики.

Разработать прототип 2D-аркады на любом игровом движке Godot Реализовать: управление спутником с клавиатуры или мыши, спавн частиц солнечного ветра с разными свойствами, систему классификации частиц по типам (захват нужным детектором), подсчёт очков и уровней сложности, простое меню и экран окончания игры.

Краткая справка о солнечном ветре и типах частиц (для игрового контекста); набор простых 2D-спрайтов (спутник, частицы трёх типов, фон звёздного неба); геймдизайн-документ с описанием механик и баланса; примеры похожих простых 2D-игр (Arkanoid, Space Invaders) как референс; базовые уроки по работе с 2D в выбранном движке.

Работающий прототип 2D-аркады, который можно запустить на обычном ПК; исходный код проекта с комментариями; краткое описание реализованных механик и логики игры; короткая презентация с демонстрацией геймплея и объяснением, как игра решает задачу популяризации науки.

Соответствие решению поставленной задаче — 40% (все заявленные механики работают, игра понятна); качество реализации и чистота кода — 25% (архитектура, отсутствие критических багов, читаемость); игровой баланс и UX — 20% (удобство управления, понятность правил, плавность геймплея); качество презентации и защиты — 15% (ясность объяснения решений и демонстрация).

№ 71. Разработать машину состояний и на ее основе реализовать анимацию игрового персонажа.

№ 72. Написать клиента для игры "выживание".

№ 73. Разработать алгоритм окружения врагами игрока.

№ 74. Кейс: Процедурная генерация космических явлений и ИИ-агенты для образовательной стратегии по солнечно-земной физике.

Институт солнечно-земной физики разрабатывает образовательную компьютерную игру в жанре стратегии, в которой игрок управляет исследовательским центром, изучающим влияние солнечной активности на Землю. В игре необходимо моделировать солнечные вспышки, корональные выбросы массы и их воздействие на магнитосферу, ионосферу и инфраструктуру Земли. Студенты выступают в роли разработчиков игровых систем.

Солнечная активность непредсказуема и многофакторна: каждая вспышка уникальна по интенсивности, направлению и составу. Предзаданные сценарии быстро надоедают игроку и не отражают реальную стохастическую природу явлений. Также необходимо, чтобы виртуальные «научные сотрудники» (ИИ-агенты) реалистично реагировали на события,

принимали решения и взаимодействовали с игроком. Требуются система процедурной генерации и интеллектуальное поведение NPC.

Разработать систему процедурной генерации солнечных событий (вспышки классов А–Х, корональные выбросы, потоки солнечного ветра) с параметрами, основанными на реальных статистических распределениях. Реализовать ИИ-агентов (научных сотрудников обсерватории), использующих деревья поведения (Behavior Trees) или конечные автоматы, которые анализируют сгенерированные события, выдают рекомендации игроку и реагируют на его команды. Интегрировать обе системы в игровой прототип.

Датасет реальных солнечных событий (GOES, NOAA) для калибровки генератора; описание физических параметров явлений; шаблоны деревьев поведения и FSM; игровой прототип с базовой архитектурой; геймдизайн-документ с описанием ролей NPC и игрового цикла; API игрового движка для работы с procedural generation и AI.

Модуль процедурной генерации с настраиваемыми параметрами распределений; система ИИ-агентов с визуализацией логики принятия решений; интегрированный прототип с демонстрацией взаимодействия агентов и генерируемых событий; отчет с описанием алгоритмов и обоснованием выбора архитектуры ИИ; презентация.

Качество процедурной генерации — 30% (разнообразие, реалистичность, параметризуемость); архитектура и поведение ИИ-агентов — 30% (логичность, адаптивность, читаемость кода); интеграция и работоспособность прототипа — 25% (отсутствие критических багов, согласованность систем); качество презентации и защиты — 15%.

№ 75. Написать генератор лабиринта.

2.3.8. Практические задания для оценки компетенции «ПК-6 ИП-2.1»

№ 76. Разработать алгоритм движения игрового объекта по случайному закону.

№ 77. Разработать игру Ping-Pong.

№ 78. Написать игру "Flappy Bird".

№ 79. При помощи системы частиц создать эффект огня.

№ 80. Кейс: Разработка 2D-аркады «Перехват солнечного ветра» для популяризации гелиофизики.

Научный институт солнечно-земной физики готовит выставку для школьников и абитуриентов и заказывает небольшую 2D-игру для интерактивного киоска. Игра должна в увлекательной форме показать, как работает система мониторинга космической погоды: игрок управляет орбитальным спутником-детектором, который перехватывает частицы солнечного ветра и классифицирует их по типам. Студенты выступают в роли начинающих разработчиков игр.

На выставке посетителям сложно объяснить, чем занимаются учёные-гелиофизики, с помощью одних лишь плакатов. Нужна простая, но наглядная игра, которая за 2–3 минуты объяснит базовые понятия: солнечный ветер, типы частиц (протоны, электроны, альфа-частицы), роль спутников-мониторов. Игра должна быть лёгкой в освоении, работать на обычном ПК и не требовать сложной графики.

Разработать прототип 2D-аркады на любом игровом движке Godot Реализовать: управление

спутником с клавиатуры или мыши, спавн частиц солнечного ветра с разными свойствами, систему классификации частиц по типам (захват нужным детектором), подсчёт очков и уровней сложности, простое меню и экран окончания игры.

Краткая справка о солнечном ветре и типах частиц (для игрового контекста); набор простых 2D-спрайтов (спутник, частицы трёх типов, фон звёздного неба); геймдизайн-документ с описанием механик и баланса; примеры похожих простых 2D-игр (Arkanoid, Space Invaders) как референс; базовые уроки по работе с 2D в выбранном движке.

Работающий прототип 2D-аркады, который можно запустить на обычном ПК; исходный код проекта с комментариями; краткое описание реализованных механик и логики игры; короткая презентация с демонстрацией геймплея и объяснением, как игра решает задачу популяризации науки.

Соответствие решению поставленной задаче — 40% (все заявленные механики работают, игра понятна); качество реализации и чистота кода — 25% (архитектура, отсутствие критических багов, читаемость); игровой баланс и UX — 20% (удобство управления, понятность правил, плавность геймплея); качество презентации и защиты — 15% (ясность объяснения решений и демонстрация).

№ 81. Разработать машину состояний и на ее основе реализовать анимацию игрового персонажа.

№ 82. Написать клиента для игры "выживание".

№ 83. Кейс: Мультиплеерная кооперативная игра для совместного управления сетью радаров.

Институт располагает распределённой сетью из нескольких когерентных радаров, разнесённых географически. Для полноценного мониторинга ионосферы операторы радаров должны координировать свои действия в реальном времени. Институт заказывает разработку кооперативной мультиплеерной игры-тренажёра, где команда из 3–5 игроков совместно управляет виртуальной радарной сетью, реагируя на геомагнитные возмущения. Студенты — команда сетевых разработчиков.

Реальные операторы радаров работают удалённо и нуждаются в отработке навыков командного взаимодействия в стрессовых ситуациях (геомагнитные бури). Существующие тренажёры рассчитаны на одного пользователя и не моделируют сетевое взаимодействие. Разработка мультиплеера сопряжена с проблемами синхронизации состояния игры, задержек сети (latency), разрешения конфликтов при одновременных действиях игроков и обеспечения плавного игрового опыта.

Разработать прототип кооперативной мультиплеерной игры с сетевой архитектурой (клиент-сервер или peer-to-peer с авторитетным сервером). Реализовать: синхронизацию состояния радаров между игроками, систему ролей (каждый игрок управляет своим радаром), механику совместного покрытия зон наблюдения, механизм компенсации сетевых задержек (lag compensation / client-side prediction), систему голосовой или текстовой коммуникации между игроками, сценарии геомагнитных бурь с нарастающей сложностью.

Карта размещения реальных радаров с параметрами (дальность, угол обзора, частота); сценарии геомагнитных возмущений (таймлайны событий); библиотека сетевых

фреймворков для выбранного движка (Mirror, Photon, Netcode for GameObjects и др.); требования к допустимой задержке (<100 мс); тестовые сценарии кооперативного геймплея; описание UX-паттернов для кооперативных интерфейсов.

Работающий сетевой прототип с подключением 3–5 игроков; демонстрация синхронизированного управления радаром в реальном времени; визуализация зон покрытия и их объединения; стресс-тест с искусственно добавленной задержкой и отчёт о стабильности; документация по сетевой архитектуре и выбору модели синхронизации; презентация.

Сетевая архитектура и синхронизация — 35% (стабильность, корректность, устойчивость к задержкам); игровой дизайн и кооперативные механики — 25% (вовлечённость, баланс, необходимость командного взаимодействия); качество кода и масштабируемость — 25% (чистота архитектуры, возможность добавления новых радаров и сценариев); качество презентации и защиты — 15%.

№ 84. Разработать алгоритм окружения врагами игрока.

№ 85. Написать генератор лабиринта.

2.3.9. Практические задания для оценки компетенции «ПК-6 ИП-2.2»

№ 86. Разработать алгоритм движения игрового объекта по случайному закону.

№ 87. Разработать игру Ping-Pong.

№ 88. Написать игру "Flappy Bird".

№ 89. При помощи системы частиц создать эффект огня.

№ 90. Кейс: Разработка 2D-аркады «Перехват солнечного ветра» для популяризации гелиофизики.

Научный институт солнечно-земной физики готовит выставку для школьников и абитуриентов и заказывает небольшую 2D-игру для интерактивного киоска. Игра должна в увлекательной форме показать, как работает система мониторинга космической погоды: игрок управляет орбитальным спутником-детектором, который перехватывает частицы солнечного ветра и классифицирует их по типам. Студенты выступают в роли начинающих разработчиков игр.

На выставке посетителям сложно объяснить, чем занимаются учёные-гелиофизики, с помощью одних лишь плакатов. Нужна простая, но наглядная игра, которая за 2–3 минуты объяснит базовые понятия: солнечный ветер, типы частиц (протоны, электроны, альфа-частицы), роль спутников-мониторов. Игра должна быть лёгкой в освоении, работать на обычном ПК и не требовать сложной графики.

Разработать прототип 2D-аркады на любом игровом движке Godot Реализовать: управление спутником с клавиатуры или мыши, спавн частиц солнечного ветра с разными свойствами, систему классификации частиц по типам (захват нужным детектором), подсчёт очков и уровней сложности, простое меню и экран окончания игры.

Краткая справка о солнечном ветре и типах частиц (для игрового контекста); набор простых 2D-спрайтов (спутник, частицы трёх типов, фон звёздного неба); геймдизайн-документ с описанием механик и баланса; примеры похожих простых 2D-игр (Arkanoid, Space Invaders) как референс; базовые уроки по работе с 2D в выбранном движке.

Работающий прототип 2D-аркады, который можно запустить на обычном ПК; исходный код проекта с комментариями; краткое описание реализованных механик и логики игры; короткая презентация с демонстрацией геймплея и объяснением, как игра решает задачу популяризации науки.

Соответствие решению поставленной задаче — 40% (все заявленные механики работают, игра понятна); качество реализации и чистота кода — 25% (архитектура, отсутствие критических багов, читаемость); игровой баланс и UX — 20% (удобство управления, понятность правил, плавность геймплея); качество презентации и защиты — 15% (ясность объяснения решений и демонстрация).

№ 91. Разработать машину состояний и на ее основе реализовать анимацию игрового персонажа.

№ 92. Написать клиента для игры "выживание".

№ 93. Разработать алгоритм окружения врагами игрока.

№ 94. Написать генератор лабиринта.

2.3.10. Практические задания для оценки компетенции «ПК-8 ППК-Р7.2»

№ 95. Разработать алгоритм движения игрового объекта по случайному закону.

№ 96. Разработать игру Ping-Pong.

№ 97. Написать игру "Flappy Bird".

№ 98. При помощи системы частиц создать эффект огня.

№ 99. Кейс: Разработка 2D-аркады «Перехват солнечного ветра» для популяризации гелиофизики.

Научный институт солнечно-земной физики готовит выставку для школьников и абитуриентов и заказывает небольшую 2D-игру для интерактивного киоска. Игра должна в увлекательной форме показать, как работает система мониторинга космической погоды: игрок управляет орбитальным спутником-детектором, который перехватывает частицы солнечного ветра и классифицирует их по типам. Студенты выступают в роли начинающих разработчиков игр.

На выставке посетителям сложно объяснить, чем занимаются учёные-гелиофизики, с помощью одних лишь плакатов. Нужна простая, но наглядная игра, которая за 2–3 минуты объяснит базовые понятия: солнечный ветер, типы частиц (протоны, электроны, альфа-частицы), роль спутников-мониторов. Игра должна быть лёгкой в освоении, работать на обычном ПК и не требовать сложной графики.

Разработать прототип 2D-аркады на любом игровом движке Godot Реализовать: управление спутником с клавиатуры или мыши, спавн частиц солнечного ветра с разными свойствами, систему классификации частиц по типам (захват нужным детектором), подсчёт очков и уровней сложности, простое меню и экран окончания игры.

Краткая справка о солнечном ветре и типах частиц (для игрового контекста); набор простых 2D-спрайтов (спутник, частицы трёх типов, фон звёздного неба); геймдизайн-документ с описанием механик и баланса; примеры похожих простых 2D-игр (Arkanoid, Space Invaders) как референс; базовые уроки по работе с 2D в выбранном движке.

Работающий прототип 2D-аркады, который можно запустить на обычном ПК; исходный код

проекта с комментариями; краткое описание реализованных механик и логики игры; короткая презентация с демонстрацией геймплея и объяснением, как игра решает задачу популяризации науки.

Соответствие решению поставленной задаче — 40% (все заявленные механики работают, игра понятна); качество реализации и чистота кода — 25% (архитектура, отсутствие критических багов, читаемость); игровой баланс и UX — 20% (удобство управления, понятность правил, плавность геймплея); качество презентации и защиты — 15% (ясность объяснения решений и демонстрация).

№ 100. Кейс: Мультиплеерная кооперативная игра для совместного управления сетью радаров.

Институт располагает распределённой сетью из нескольких когерентных радаров, разнесённых географически. Для полноценного мониторинга ионосферы операторы радаров должны координировать свои действия в реальном времени. Институт заказывает разработку кооперативной мультиплеерной игры-тренажёра, где команда из 3–5 игроков совместно управляет виртуальной радарной сетью, реагируя на геомагнитные возмущения. Студенты — команда сетевых разработчиков.

Реальные операторы радаров работают удалённо и нуждаются в отработке навыков командного взаимодействия в стрессовых ситуациях (геомагнитные бури). Существующие тренажёры рассчитаны на одного пользователя и не моделируют сетевое взаимодействие. Разработка мультиплеера сопряжена с проблемами синхронизации состояния игры, задержек сети (latency), разрешения конфликтов при одновременных действиях игроков и обеспечения плавного игрового опыта.

Разработать прототип кооперативной мультиплеерной игры с сетевой архитектурой (клиент-сервер или peer-to-peer с авторитетным сервером). Реализовать: синхронизацию состояния радаров между игроками, систему ролей (каждый игрок управляет своим радаром), механику совместного покрытия зон наблюдения, механизм компенсации сетевых задержек (lag compensation / client-side prediction), систему голосовой или текстовой коммуникации между игроками, сценарии геомагнитных бурь с нарастающей сложностью.

Карта размещения реальных радаров с параметрами (дальность, угол обзора, частота); сценарии геомагнитных возмущений (таймлайны событий); библиотека сетевых фреймворков для выбранного движка (Mirror, Photon, Netcode for GameObjects и др.); требования к допустимой задержке (<100 мс); тестовые сценарии кооперативного геймплея; описание UX-паттернов для кооперативных интерфейсов.

Работающий сетевой прототип с подключением 3–5 игроков; демонстрация синхронизированного управления радаром в реальном времени; визуализация зон покрытия и их объединения; стресс-тест с искусственно добавленной задержкой и отчёт о стабильности; документация по сетевой архитектуре и выбору модели синхронизации; презентация.

Сетевая архитектура и синхронизация — 35% (стабильность, корректность, устойчивость к задержкам); игровой дизайн и кооперативные механики — 25% (вовлечённость, баланс, необходимость командного взаимодействия); качество кода и масштабируемость — 25% (чистота архитектуры, возможность добавления новых радаров и сценариев); качество презентации и защиты — 15%.

№ 101. Кейс: Процедурная генерация космических явлений и ИИ-агенты для образовательной стратегии по солнечно-земной физике.

Институт солнечно-земной физики разрабатывает образовательную компьютерную игру в жанре стратегии, в которой игрок управляет исследовательским центром, изучающим влияние солнечной активности на Землю. В игре необходимо моделировать солнечные вспышки, корональные выбросы массы и их воздействие на магнитосферу, ионосферу и инфраструктуру Земли. Студенты выступают в роли разработчиков игровых систем.

Солнечная активность непредсказуема и многофакторна: каждая вспышка уникальна по интенсивности, направлению и составу. Предзаданные сценарии быстро надоедают игроку и не отражают реальную стохастическую природу явлений. Также необходимо, чтобы виртуальные «научные сотрудники» (ИИ-агенты) реалистично реагировали на события, принимали решения и взаимодействовали с игроком. Требуются система процедурной генерации и интеллектуальное поведение NPC.

Разработать систему процедурной генерации солнечных событий (вспышки классов А–Х, корональные выбросы, потоки солнечного ветра) с параметрами, основанными на реальных статистических распределениях. Реализовать ИИ-агентов (научных сотрудников обсерватории), использующих деревья поведения (Behavior Trees) или конечные автоматы, которые анализируют сгенерированные события, выдают рекомендации игроку и реагируют на его команды. Интегрировать обе системы в игровой прототип.

Датасет реальных солнечных событий (GOES, NOAA) для калибровки генератора; описание физических параметров явлений; шаблоны деревьев поведения и FSM; игровой прототип с базовой архитектурой; геймдизайн-документ с описанием ролей NPC и игрового цикла; API игрового движка для работы с procedural generation и AI.

Модуль процедурной генерации с настраиваемыми параметрами распределений; система ИИ-агентов с визуализацией логики принятия решений; интегрированный прототип с демонстрацией взаимодействия агентов и генерируемых событий; отчет с описанием алгоритмов и обоснованием выбора архитектуры ИИ; презентация.

Качество процедурной генерации — 30% (разнообразие, реалистичность, параметризуемость); архитектура и поведение ИИ-агентов — 30% (логичность, адаптивность, читаемость кода); интеграция и работоспособность прототипа — 25% (отсутствие критических багов, согласованность систем); качество презентации и защиты — 15%.

№ 102. Написать генератор лабиринта.

2.3.11. Практические задания для оценки компетенции «ПК-1 ППК-Р1.4»

№ 103. Разработать алгоритм движения игрового объекта по случайному закону.

№ 104. Основные конструкции языка и их применение.

№ 105. Разработать игру Ping-Pong.

№ 106. Написать игру "Flappy Bird".

№ 107. Кейс: Разработка 2D-аркады «Перехват солнечного ветра» для популяризации гелиофизики.

Научный институт солнечно-земной физики готовит выставку для школьников и абитуриентов и заказывает небольшую 2D-игру для интерактивного киоска. Игра должна в

увлекательной форме показать, как работает система мониторинга космической погоды: игрок управляет орбитальным спутником-детектором, который перехватывает частицы солнечного ветра и классифицирует их по типам. Студенты выступают в роли начинающих разработчиков игр.

На выставке посетителям сложно объяснить, чем занимаются учёные-гелиофизики, с помощью одних лишь плакатов. Нужна простая, но наглядная игра, которая за 2–3 минуты объяснит базовые понятия: солнечный ветер, типы частиц (протоны, электроны, альфа-частицы), роль спутников-мониторов. Игра должна быть лёгкой в освоении, работать на обычном ПК и не требовать сложной графики.

Разработать прототип 2D-аркады на любом игровом движке Godot Реализовать: управление спутником с клавиатуры или мыши, спавн частиц солнечного ветра с разными свойствами, систему классификации частиц по типам (захват нужным детектором), подсчёт очков и уровней сложности, простое меню и экран окончания игры.

Краткая справка о солнечном ветре и типах частиц (для игрового контекста); набор простых 2D-спрайтов (спутник, частицы трёх типов, фон звёздного неба); геймдизайн-документ с описанием механик и баланса; примеры похожих простых 2D-игр (Arkanoid, Space Invaders) как референс; базовые уроки по работе с 2D в выбранном движке.

Работающий прототип 2D-аркады, который можно запустить на обычном ПК; исходный код проекта с комментариями; краткое описание реализованных механик и логики игры; короткая презентация с демонстрацией геймплея и объяснением, как игра решает задачу популяризации науки.

Соответствие решению поставленной задаче — 40% (все заявленные механики работают, игра понятна); качество реализации и чистота кода — 25% (архитектура, отсутствие критических багов, читаемость); игровой баланс и UX — 20% (удобство управления, понятность правил, плавность геймплея); качество презентации и защиты — 15% (ясность объяснения решений и демонстрация).

№ 108. Разработать машину состояний и на ее основе реализовать анимацию игрового персонажа.

№ 109. Написать клиента для игры "выживание".

№ 110. Кейс: Мультиплеерная кооперативная игра для совместного управления сетью радаров.

Институт располагает распределённой сетью из нескольких когерентных радаров, разнесённых географически. Для полноценного мониторинга ионосферы операторы радаров должны координировать свои действия в реальном времени. Институт заказывает разработку кооперативной мультиплеерной игры-тренажёра, где команда из 3–5 игроков совместно управляет виртуальной радарной сетью, реагируя на геомагнитные возмущения. Студенты — команда сетевых разработчиков.

Реальные операторы радаров работают удалённо и нуждаются в отработке навыков командного взаимодействия в стрессовых ситуациях (геомагнитные бури). Существующие тренажёры рассчитаны на одного пользователя и не моделируют сетевое взаимодействие. Разработка мультиплеера сопряжена с проблемами синхронизации состояния игры, задержек

сети (latency), разрешения конфликтов при одновременных действиях игроков и обеспечения плавного игрового опыта.

Разработать прототип кооперативной мультиплеерной игры с сетевой архитектурой (клиент-сервер или peer-to-peer с авторитетным сервером). Реализовать: синхронизацию состояния радаров между игроками, систему ролей (каждый игрок управляет своим радаром), механику совместного покрытия зон наблюдения, механизм компенсации сетевых задержек (lag compensation / client-side prediction), систему голосовой или текстовой коммуникации между игроками, сценарии геомагнитных бурь с нарастающей сложностью.

Карта размещения реальных радаров с параметрами (дальность, угол обзора, частота); сценарии геомагнитных возмущений (таймлайны событий); библиотека сетевых фреймворков для выбранного движка (Mirror, Photon, Netcode for GameObjects и др.); требования к допустимой задержке (<100 мс); тестовые сценарии кооперативного геймплея; описание UX-паттернов для кооперативных интерфейсов.

Работающий сетевой прототип с подключением 3–5 игроков; демонстрация синхронизированного управления радаром в реальном времени; визуализация зон покрытия и их объединения; стресс-тест с искусственно добавленной задержкой и отчёт о стабильности; документация по сетевой архитектуре и выбору модели синхронизации; презентация.

Сетевая архитектура и синхронизация — 35% (стабильность, корректность, устойчивость к задержкам); игровой дизайн и кооперативные механики — 25% (вовлечённость, баланс, необходимость командного взаимодействия); качество кода и масштабируемость — 25% (чистота архитектуры, возможность добавления новых радаров и сценариев); качество презентации и защиты — 15%.

№ 111. Разработать алгоритм окружения врагами игрока.

№ 112. Кейс: Процедурная генерация космических явлений и ИИ-агенты для образовательной стратегии по солнечно-земной физике.

Институт солнечно-земной физики разрабатывает образовательную компьютерную игру в жанре стратегии, в которой игрок управляет исследовательским центром, изучающим влияние солнечной активности на Землю. В игре необходимо моделировать солнечные вспышки, корональные выбросы массы и их воздействие на магнитосферу, ионосферу и инфраструктуру Земли. Студенты выступают в роли разработчиков игровых систем.

Солнечная активность непредсказуема и многофакторна: каждая вспышка уникальна по интенсивности, направлению и составу. Предзаданные сценарии быстро надоедают игроку и не отражают реальную стохастическую природу явлений. Также необходимо, чтобы виртуальные «научные сотрудники» (ИИ-агенты) реалистично реагировали на события, принимали решения и взаимодействовали с игроком. Требуются система процедурной генерации и интеллектуальное поведение NPC.

Разработать систему процедурной генерации солнечных событий (вспышки классов А–Х, корональные выбросы, потоки солнечного ветра) с параметрами, основанными на реальных статистических распределениях. Реализовать ИИ-агентов (научных сотрудников обсерватории), использующих деревья поведения (Behavior Trees) или конечные автоматы, которые анализируют сгенерированные события, выдают рекомендации игроку и реагируют на его команды. Интегрировать обе системы в игровой прототип.

Датасет реальных солнечных событий (GOES, NOAA) для калибровки генератора; описание физических параметров явлений; шаблоны деревьев поведения и FSM; игровой прототип с базовой архитектурой; геймдизайн-документ с описанием ролей NPC и игрового цикла; API игрового движка для работы с procedural generation и AI.

Модуль процедурной генерации с настраиваемыми параметрами распределений; система ИИ-агентов с визуализацией логики принятия решений; интегрированный прототип с демонстрацией взаимодействия агентов и генерируемых событий; отчет с описанием алгоритмов и обоснованием выбора архитектуры ИИ; презентация.

Качество процедурной генерации — 30% (разнообразие, реалистичность, параметризуемость); архитектура и поведение ИИ-агентов — 30% (логичность, адаптивность, читаемость кода); интеграция и работоспособность прототипа — 25% (отсутствие критических багов, согласованность систем); качество презентации и защиты — 15%.

№ 113. Написать генератор лабиринта.

2.3.12. Практические задания для оценки компетенции «ПК-1 ППК-Р1.5»

№ 114. Разработать алгоритм движения игрового объекта по случайному закону.

№ 115. Основные конструкции языка и их применение.

№ 116. Разработать игру Ping-Pong.

№ 117. Написать игру "Flappy Bird".

№ 118. Кейс: Разработка 2D-аркады «Перехват солнечного ветра» для популяризации гелиофизики.

Научный институт солнечно-земной физики готовит выставку для школьников и абитуриентов и заказывает небольшую 2D-игру для интерактивного киоска. Игра должна в увлекательной форме показать, как работает система мониторинга космической погоды: игрок управляет орбитальным спутником-детектором, который перехватывает частицы солнечного ветра и классифицирует их по типам. Студенты выступают в роли начинающих разработчиков игр.

На выставке посетителям сложно объяснить, чем занимаются учёные-гелиофизики, с помощью одних лишь плакатов. Нужна простая, но наглядная игра, которая за 2–3 минуты объяснит базовые понятия: солнечный ветер, типы частиц (протоны, электроны, альфа-частицы), роль спутников-мониторов. Игра должна быть лёгкой в освоении, работать на обычном ПК и не требовать сложной графики.

Разработать прототип 2D-аркады на любом игровом движке Godot Реализовать: управление спутником с клавиатуры или мыши, спавн частиц солнечного ветра с разными свойствами, систему классификации частиц по типам (захват нужным детектором), подсчёт очков и уровней сложности, простое меню и экран окончания игры.

Краткая справка о солнечном ветре и типах частиц (для игрового контекста); набор простых 2D-спрайтов (спутник, частицы трёх типов, фон звёздного неба); геймдизайн-документ с описанием механик и баланса; примеры похожих простых 2D-игр (Arkanoid, Space Invaders) как референс; базовые уроки по работе с 2D в выбранном движке.

Работающий прототип 2D-аркады, который можно запустить на обычном ПК; исходный код

проекта с комментариями; краткое описание реализованных механик и логики игры; короткая презентация с демонстрацией геймплея и объяснением, как игра решает задачу популяризации науки.

Соответствие решению поставленной задаче — 40% (все заявленные механики работают, игра понятна); качество реализации и чистота кода — 25% (архитектура, отсутствие критических багов, читаемость); игровой баланс и UX — 20% (удобство управления, понятность правил, плавность геймплея); качество презентации и защиты — 15% (ясность объяснения решений и демонстрация).

№ 119. Разработать машину состояний и на ее основе реализовать анимацию игрового персонажа.

№ 120. Написать клиента для игры "выживание".

№ 121. Кейс: Мультиплеерная кооперативная игра для совместного управления сетью радаров.

Институт располагает распределённой сетью из нескольких когерентных радаров, разнесённых географически. Для полноценного мониторинга ионосферы операторы радаров должны координировать свои действия в реальном времени. Институт заказывает разработку кооперативной мультиплеерной игры-тренажёра, где команда из 3–5 игроков совместно управляет виртуальной радарной сетью, реагируя на геомагнитные возмущения. Студенты — команда сетевых разработчиков.

Реальные операторы радаров работают удалённо и нуждаются в отработке навыков командного взаимодействия в стрессовых ситуациях (геомагнитные бури). Существующие тренажёры рассчитаны на одного пользователя и не моделируют сетевое взаимодействие. Разработка мультиплеера сопряжена с проблемами синхронизации состояния игры, задержек сети (latency), разрешения конфликтов при одновременных действиях игроков и обеспечения плавного игрового опыта.

Разработать прототип кооперативной мультиплеерной игры с сетевой архитектурой (клиент-сервер или peer-to-peer с авторитетным сервером). Реализовать: синхронизацию состояния радаров между игроками, систему ролей (каждый игрок управляет своим радаром), механику совместного покрытия зон наблюдения, механизм компенсации сетевых задержек (lag compensation / client-side prediction), систему голосовой или текстовой коммуникации между игроками, сценарии геомагнитных бурь с нарастающей сложностью.

Карта размещения реальных радаров с параметрами (дальность, угол обзора, частота); сценарии геомагнитных возмущений (таймлайны событий); библиотека сетевых фреймворков для выбранного движка (Mirror, Photon, Netcode for GameObjects и др.); требования к допустимой задержке (<100 мс); тестовые сценарии кооперативного геймплея; описание UX-паттернов для кооперативных интерфейсов.

Работающий сетевой прототип с подключением 3–5 игроков; демонстрация синхронизированного управления радаром в реальном времени; визуализация зон покрытия и их объединения; стресс-тест с искусственно добавленной задержкой и отчёт о стабильности; документация по сетевой архитектуре и выбору модели синхронизации; презентация.

Сетевая архитектура и синхронизация — 35% (стабильность, корректность, устойчивость к

задержкам); игровой дизайн и кооперативные механики — 25% (вовлечённость, баланс, необходимость командного взаимодействия); качество кода и масштабируемость — 25% (чистота архитектуры, возможность добавления новых радаров и сценариев); качество презентации и защиты — 15%.

№ 122. Разработать алгоритм окружения врагами игрока.

№ 123. Кейс: Процедурная генерация космических явлений и ИИ-агенты для образовательной стратегии по солнечно-земной физике.

Институт солнечно-земной физики разрабатывает образовательную компьютерную игру в жанре стратегии, в которой игрок управляет исследовательским центром, изучающим влияние солнечной активности на Землю. В игре необходимо моделировать солнечные вспышки, корональные выбросы массы и их воздействие на магнитосферу, ионосферу и инфраструктуру Земли. Студенты выступают в роли разработчиков игровых систем.

Солнечная активность непредсказуема и многофакторна: каждая вспышка уникальна по интенсивности, направлению и составу. Предзаданные сценарии быстро надоедают игроку и не отражают реальную стохастическую природу явлений. Также необходимо, чтобы виртуальные «научные сотрудники» (ИИ-агенты) реалистично реагировали на события, принимали решения и взаимодействовали с игроком. Требуются система процедурной генерации и интеллектуальное поведение NPC.

Разработать систему процедурной генерации солнечных событий (вспышки классов А–Х, корональные выбросы, потоки солнечного ветра) с параметрами, основанными на реальных статистических распределениях. Реализовать ИИ-агентов (научных сотрудников обсерватории), использующих деревья поведения (Behavior Trees) или конечные автоматы, которые анализируют сгенерированные события, выдают рекомендации игроку и реагируют на его команды. Интегрировать обе системы в игровой прототип.

Датасет реальных солнечных событий (GOES, NOAA) для калибровки генератора; описание физических параметров явлений; шаблоны деревьев поведения и FSM; игровой прототип с базовой архитектурой; геймдизайн-документ с описанием ролей NPC и игрового цикла; API игрового движка для работы с procedural generation и AI.

Модуль процедурной генерации с настраиваемыми параметрами распределений; система ИИ-агентов с визуализацией логики принятия решений; интегрированный прототип с демонстрацией взаимодействия агентов и генерируемых событий; отчет с описанием алгоритмов и обоснованием выбора архитектуры ИИ; презентация.

Качество процедурной генерации — 30% (разнообразие, реалистичность, параметризуемость); архитектура и поведение ИИ-агентов — 30% (логичность, адаптивность, читаемость кода); интеграция и работоспособность прототипа — 25% (отсутствие критических багов, согласованность систем); качество презентации и защиты — 15%.

№ 124. Написать генератор лабиринта.

2.3.13. Практические задания для оценки компетенции «ПК-2 ППК-Р2.4»

№ 125. Разработать алгоритм движения игрового объекта по случайному закону.

№ 126. Основные конструкции языка и их применение.

№ 127. Разработать игру Ping-Pong.

№ 128. Написать игру "Flappy Bird".

№ 129. Кейс: Разработка 2D-аркады «Перехват солнечного ветра» для популяризации гелиофизики.

Научный институт солнечно-земной физики готовит выставку для школьников и абитуриентов и заказывает небольшую 2D-игру для интерактивного киоска. Игра должна в увлекательной форме показать, как работает система мониторинга космической погоды: игрок управляет орбитальным спутником-детектором, который перехватывает частицы солнечного ветра и классифицирует их по типам. Студенты выступают в роли начинающих разработчиков игр.

На выставке посетителям сложно объяснить, чем занимаются учёные-гелиофизики, с помощью одних лишь плакатов. Нужна простая, но наглядная игра, которая за 2–3 минуты объяснит базовые понятия: солнечный ветер, типы частиц (протоны, электроны, альфа-частицы), роль спутников-мониторов. Игра должна быть лёгкой в освоении, работать на обычном ПК и не требовать сложной графики.

Разработать прототип 2D-аркады на любом игровом движке Godot Реализовать: управление спутником с клавиатуры или мыши, спавн частиц солнечного ветра с разными свойствами, систему классификации частиц по типам (захват нужным детектором), подсчёт очков и уровней сложности, простое меню и экран окончания игры.

Краткая справка о солнечном ветре и типах частиц (для игрового контекста); набор простых 2D-спрайтов (спутник, частицы трёх типов, фон звёздного неба); геймдизайн-документ с описанием механик и баланса; примеры похожих простых 2D-игр (Arkanoid, Space Invaders) как референс; базовые уроки по работе с 2D в выбранном движке.

Работающий прототип 2D-аркады, который можно запустить на обычном ПК; исходный код проекта с комментариями; краткое описание реализованных механик и логики игры; короткая презентация с демонстрацией геймплея и объяснением, как игра решает задачу популяризации науки.

Соответствие решению поставленной задаче — 40% (все заявленные механики работают, игра понятна); качество реализации и чистота кода — 25% (архитектура, отсутствие критических багов, читаемость); игровой баланс и UX — 20% (удобство управления, понятность правил, плавность геймплея); качество презентации и защиты — 15% (ясность объяснения решений и демонстрация).

№ 130. Разработать машину состояний и на ее основе реализовать анимацию игрового персонажа.

№ 131. Разработать алгоритм окружения врагами игрока.

№ 132. Кейс: Процедурная генерация космических явлений и ИИ-агенты для образовательной стратегии по солнечно-земной физике.

Институт солнечно-земной физики разрабатывает образовательную компьютерную игру в жанре стратегии, в которой игрок управляет исследовательским центром, изучающим влияние солнечной активности на Землю. В игре необходимо моделировать солнечные вспышки, корональные выбросы массы и их воздействие на магнитосферу, ионосферу и инфраструктуру Земли. Студенты выступают в роли разработчиков игровых систем.

Солнечная активность непредсказуема и многофакторна: каждая вспышка уникальна по интенсивности, направлению и составу. Предзаданные сценарии быстро надоедают игроку и не отражают реальную стохастическую природу явлений. Также необходимо, чтобы виртуальные «научные сотрудники» (ИИ-агенты) реалистично реагировали на события, принимали решения и взаимодействовали с игроком. Требуются система процедурной генерации и интеллектуальное поведение NPC.

Разработать систему процедурной генерации солнечных событий (вспышки классов А–Х, корональные выбросы, потоки солнечного ветра) с параметрами, основанными на реальных статистических распределениях. Реализовать ИИ-агентов (научных сотрудников обсерватории), использующих деревья поведения (Behavior Trees) или конечные автоматы, которые анализируют сгенерированные события, выдают рекомендации игроку и реагируют на его команды. Интегрировать обе системы в игровой прототип.

Датасет реальных солнечных событий (GOES, NOAA) для калибровки генератора; описание физических параметров явлений; шаблоны деревьев поведения и FSM; игровой прототип с базовой архитектурой; геймдизайн-документ с описанием ролей NPC и игрового цикла; API игрового движка для работы с procedural generation и AI.

Модуль процедурной генерации с настраиваемыми параметрами распределений; система ИИ-агентов с визуализацией логики принятия решений; интегрированный прототип с демонстрацией взаимодействия агентов и генерируемых событий; отчет с описанием алгоритмов и обоснованием выбора архитектуры ИИ; презентация.

Качество процедурной генерации — 30% (разнообразие, реалистичность, параметризуемость); архитектура и поведение ИИ-агентов — 30% (логичность, адаптивность, читаемость кода); интеграция и работоспособность прототипа — 25% (отсутствие критических багов, согласованность систем); качество презентации и защиты — 15%.

№ 133. Написать генератор лабиринта.

2.3.14. Практические задания для оценки компетенции «ПК-8 ППК-Р7.3»

№ 134. Разработать алгоритм движения игрового объекта по случайному закону.

№ 135. Разработать игру Ping-Pong.

№ 136. Написать игру "Flappy Bird".

№ 137. Кейс: Разработка 2D-аркады «Перехват солнечного ветра» для популяризации гелиофизики.

Научный институт солнечно-земной физики готовит выставку для школьников и абитуриентов и заказывает небольшую 2D-игру для интерактивного киоска. Игра должна в увлекательной форме показать, как работает система мониторинга космической погоды: игрок управляет орбитальным спутником-детектором, который перехватывает частицы солнечного ветра и классифицирует их по типам. Студенты выступают в роли начинающих разработчиков игр.

На выставке посетителям сложно объяснить, чем занимаются учёные-гелиофизики, с помощью одних лишь плакатов. Нужна простая, но наглядная игра, которая за 2–3 минуты объяснит базовые понятия: солнечный ветер, типы частиц (протоны, электроны, альфа-частицы), роль спутников-мониторов. Игра должна быть лёгкой в освоении, работать на обычном ПК и не требовать сложной графики.

Разработать прототип 2D-аркады на любом игровом движке Godot Реализовать: управление спутником с клавиатуры или мыши, спавн частиц солнечного ветра с разными свойствами, систему классификации частиц по типам (захват нужным детектором), подсчёт очков и уровней сложности, простое меню и экран окончания игры.

Краткая справка о солнечном ветре и типах частиц (для игрового контекста); набор простых 2D-спрайтов (спутник, частицы трёх типов, фон звёздного неба); геймдизайн-документ с описанием механик и баланса; примеры похожих простых 2D-игр (Arkanoid, Space Invaders) как референс; базовые уроки по работе с 2D в выбранном движке.

Работающий прототип 2D-аркады, который можно запустить на обычном ПК; исходный код проекта с комментариями; краткое описание реализованных механик и логики игры; короткая презентация с демонстрацией геймплея и объяснением, как игра решает задачу популяризации науки.

Соответствие решению поставленной задаче — 40% (все заявленные механики работают, игра понятна); качество реализации и чистота кода — 25% (архитектура, отсутствие критических багов, читаемость); игровой баланс и UX — 20% (удобство управления, понятность правил, плавность геймплея); качество презентации и защиты — 15% (ясность объяснения решений и демонстрация).

№ 138. Написать клиента для игры "выживание".

№ 139. Написать генератор лабиринта.

2.3.15. Практические задания для оценки компетенции «ПК-1 ППК-Р1.6»

№ 140. Разработать алгоритм движения игрового объекта по случайному закону.

№ 141. Разработать игру Ping-Pong.

№ 142. Написать игру "Flappy Bird".

№ 143. При помощи системы частиц создать эффект огня.

№ 144. Кейс: Разработка 2D-аркады «Перехват солнечного ветра» для популяризации гелиофизики.

Научный институт солнечно-земной физики готовит выставку для школьников и абитуриентов и заказывает небольшую 2D-игру для интерактивного киоска. Игра должна в увлекательной форме показать, как работает система мониторинга космической погоды: игрок управляет орбитальным спутником-детектором, который перехватывает частицы солнечного ветра и классифицирует их по типам. Студенты выступают в роли начинающих разработчиков игр.

На выставке посетителям сложно объяснить, чем занимаются учёные-гелиофизики, с помощью одних лишь плакатов. Нужна простая, но наглядная игра, которая за 2–3 минуты объяснит базовые понятия: солнечный ветер, типы частиц (протоны, электроны, альфа-частицы), роль спутников-мониторов. Игра должна быть лёгкой в освоении, работать на обычном ПК и не требовать сложной графики.

Разработать прототип 2D-аркады на любом игровом движке Godot Реализовать: управление спутником с клавиатуры или мыши, спавн частиц солнечного ветра с разными свойствами, систему классификации частиц по типам (захват нужным детектором), подсчёт очков и уровней сложности, простое меню и экран окончания игры.

Краткая справка о солнечном ветре и типах частиц (для игрового контекста); набор простых 2D-спрайтов (спутник, частицы трёх типов, фон звёздного неба); геймдизайн-документ с описанием механик и баланса; примеры похожих простых 2D-игр (Arkanoid, Space Invaders) как референс; базовые уроки по работе с 2D в выбранном движке.

Работающий прототип 2D-аркады, который можно запустить на обычном ПК; исходный код проекта с комментариями; краткое описание реализованных механик и логики игры; короткая презентация с демонстрацией геймплея и объяснением, как игра решает задачу популяризации науки.

Соответствие решению поставленной задаче — 40% (все заявленные механики работают, игра понятна); качество реализации и чистота кода — 25% (архитектура, отсутствие критических багов, читаемость); игровой баланс и UX — 20% (удобство управления, понятность правил, плавность геймплея); качество презентации и защиты — 15% (ясность объяснения решений и демонстрация).

№ 145. Написать клиента для игры "выживание".

№ 146. Кейс: Мультиплеерная кооперативная игра для совместного управления сетью радаров.

Институт располагает распределённой сетью из нескольких когерентных радаров, разнесённых географически. Для полноценного мониторинга ионосферы операторы радаров должны координировать свои действия в реальном времени. Институт заказывает разработку кооперативной мультиплеерной игры-тренажёра, где команда из 3–5 игроков совместно управляет виртуальной радарной сетью, реагируя на геомагнитные возмущения. Студенты — команда сетевых разработчиков.

Реальные операторы радаров работают удалённо и нуждаются в отработке навыков командного взаимодействия в стрессовых ситуациях (геомагнитные бури). Существующие тренажёры рассчитаны на одного пользователя и не моделируют сетевое взаимодействие. Разработка мультиплеера сопряжена с проблемами синхронизации состояния игры, задержек сети (latency), разрешения конфликтов при одновременных действиях игроков и обеспечения плавного игрового опыта.

Разработать прототип кооперативной мультиплеерной игры с сетевой архитектурой (клиент-сервер или peer-to-peer с авторитетным сервером). Реализовать: синхронизацию состояния радаров между игроками, систему ролей (каждый игрок управляет своим радаром), механику совместного покрытия зон наблюдения, механизм компенсации сетевых задержек (lag compensation / client-side prediction), систему голосовой или текстовой коммуникации между игроками, сценарии геомагнитных бурь с нарастающей сложностью.

Карта размещения реальных радаров с параметрами (дальность, угол обзора, частота); сценарии геомагнитных возмущений (таймлайны событий); библиотека сетевых фреймворков для выбранного движка (Mirror, Photon, Netcode for GameObjects и др.); требования к допустимой задержке (<100 мс); тестовые сценарии кооперативного геймплея; описание UX-паттернов для кооперативных интерфейсов.

Работающий сетевой прототип с подключением 3–5 игроков; демонстрация синхронизированного управления радаром в реальном времени; визуализация зон покрытия и их объединения; стресс-тест с искусственно добавленной задержкой и отчёт о стабильности;

документация по сетевой архитектуре и выбору модели синхронизации; презентация.

Сетевая архитектура и синхронизация — 35% (стабильность, корректность, устойчивость к задержкам); игровой дизайн и кооперативные механики — 25% (вовлечённость, баланс, необходимость командного взаимодействия); качество кода и масштабируемость — 25% (чистота архитектуры, возможность добавления новых радаров и сценариев); качество презентации и защиты — 15%.

№ 147. Кейс: Процедурная генерация космических явлений и ИИ-агенты для образовательной стратегии по солнечно-земной физике.

Институт солнечно-земной физики разрабатывает образовательную компьютерную игру в жанре стратегии, в которой игрок управляет исследовательским центром, изучающим влияние солнечной активности на Землю. В игре необходимо моделировать солнечные вспышки, корональные выбросы массы и их воздействие на магнитосферу, ионосферу и инфраструктуру Земли. Студенты выступают в роли разработчиков игровых систем.

Солнечная активность непредсказуема и многофакторна: каждая вспышка уникальна по интенсивности, направлению и составу. Предзаданные сценарии быстро надоедают игроку и не отражают реальную стохастическую природу явлений. Также необходимо, чтобы виртуальные «научные сотрудники» (ИИ-агенты) реалистично реагировали на события, принимали решения и взаимодействовали с игроком. Требуются система процедурной генерации и интеллектуальное поведение NPC.

Разработать систему процедурной генерации солнечных событий (вспышки классов А–Х, корональные выбросы, потоки солнечного ветра) с параметрами, основанными на реальных статистических распределениях. Реализовать ИИ-агентов (научных сотрудников обсерватории), использующих деревья поведения (Behavior Trees) или конечные автоматы, которые анализируют сгенерированные события, выдают рекомендации игроку и реагируют на его команды. Интегрировать обе системы в игровой прототип.

Датасет реальных солнечных событий (GOES, NOAA) для калибровки генератора; описание физических параметров явлений; шаблоны деревьев поведения и FSM; игровой прототип с базовой архитектурой; геймдизайн-документ с описанием ролей NPC и игрового цикла; API игрового движка для работы с procedural generation и AI.

Модуль процедурной генерации с настраиваемыми параметрами распределений; система ИИ-агентов с визуализацией логики принятия решений; интегрированный прототип с демонстрацией взаимодействия агентов и генерируемых событий; отчет с описанием алгоритмов и обоснованием выбора архитектуры ИИ; презентация.

Качество процедурной генерации — 30% (разнообразие, реалистичность, параметризуемость); архитектура и поведение ИИ-агентов — 30% (логичность, адаптивность, читаемость кода); интеграция и работоспособность прототипа — 25% (отсутствие критических багов, согласованность систем); качество презентации и защиты — 15%.

№ 148. Написать генератор лабиринта.

2.3.16. Практические задания для оценки компетенции «ПК-1 ППК-Р1.7»

№ 149. Разработать алгоритм движения игрового объекта по случайному закону.

№ 150. Основные конструкции языка и их применение.

№ 151. Разработать игру *Ping-Pong*.

№ 152. Написать игру *"Flappy Bird"*.

№ 153. Кейс: Разработка 2D-аркады «Перехват солнечного ветра» для популяризации гелиофизики.

Научный институт солнечно-земной физики готовит выставку для школьников и абитуриентов и заказывает небольшую 2D-игру для интерактивного киоска. Игра должна в увлекательной форме показать, как работает система мониторинга космической погоды: игрок управляет орбитальным спутником-детектором, который перехватывает частицы солнечного ветра и классифицирует их по типам. Студенты выступают в роли начинающих разработчиков игр.

На выставке посетителям сложно объяснить, чем занимаются учёные-гелиофизики, с помощью одних лишь плакатов. Нужна простая, но наглядная игра, которая за 2–3 минуты объяснит базовые понятия: солнечный ветер, типы частиц (протоны, электроны, альфа-частицы), роль спутников-мониторов. Игра должна быть лёгкой в освоении, работать на обычном ПК и не требовать сложной графики.

Разработать прототип 2D-аркады на любом игровом движке Godot Реализовать: управление спутником с клавиатуры или мыши, спавн частиц солнечного ветра с разными свойствами, систему классификации частиц по типам (захват нужным детектором), подсчёт очков и уровней сложности, простое меню и экран окончания игры.

Краткая справка о солнечном ветре и типах частиц (для игрового контекста); набор простых 2D-спрайтов (спутник, частицы трёх типов, фон звёздного неба); геймдизайн-документ с описанием механик и баланса; примеры похожих простых 2D-игр (Arkanoid, Space Invaders) как референс; базовые уроки по работе с 2D в выбранном движке.

Работающий прототип 2D-аркады, который можно запустить на обычном ПК; исходный код проекта с комментариями; краткое описание реализованных механик и логики игры; короткая презентация с демонстрацией геймплея и объяснением, как игра решает задачу популяризации науки.

Соответствие решению поставленной задаче — 40% (все заявленные механики работают, игра понятна); качество реализации и чистота кода — 25% (архитектура, отсутствие критических багов, читаемость); игровой баланс и UX — 20% (удобство управления, понятность правил, плавность геймплея); качество презентации и защиты — 15% (ясность объяснения решений и демонстрация).

№ 154. Написать клиента для игры *"выживание"*.

№ 155. Кейс: Мультиплеерная кооперативная игра для совместного управления сетью радаров.

Институт располагает распределённой сетью из нескольких когерентных радаров, разнесённых географически. Для полноценного мониторинга ионосферы операторы радаров должны координировать свои действия в реальном времени. Институт заказывает разработку кооперативной мультиплеерной игры-тренажёра, где команда из 3–5 игроков совместно управляет виртуальной радарной сетью, реагируя на геомагнитные возмущения. Студенты — команда сетевых разработчиков.

Реальные операторы радаров работают удалённо и нуждаются в отработке навыков командного взаимодействия в стрессовых ситуациях (геомагнитные бури). Существующие тренажёры рассчитаны на одного пользователя и не моделируют сетевое взаимодействие. Разработка мультиплеера сопряжена с проблемами синхронизации состояния игры, задержек сети (latency), разрешения конфликтов при одновременных действиях игроков и обеспечения плавного игрового опыта.

Разработать прототип кооперативной мультиплеерной игры с сетевой архитектурой (клиент-сервер или peer-to-peer с авторитетным сервером). Реализовать: синхронизацию состояния радаров между игроками, систему ролей (каждый игрок управляет своим радаром), механику совместного покрытия зон наблюдения, механизм компенсации сетевых задержек (lag compensation / client-side prediction), систему голосовой или текстовой коммуникации между игроками, сценарии геомагнитных бурь с нарастающей сложностью.

Карта размещения реальных радаров с параметрами (дальность, угол обзора, частота); сценарии геомагнитных возмущений (таймлайны событий); библиотека сетевых фреймворков для выбранного движка (Mirror, Photon, Netcode for GameObjects и др.); требования к допустимой задержке (<100 мс); тестовые сценарии кооперативного геймплея; описание UX-паттернов для кооперативных интерфейсов.

Работающий сетевой прототип с подключением 3–5 игроков; демонстрация синхронизированного управления радаром в реальном времени; визуализация зон покрытия и их объединения; стресс-тест с искусственно добавленной задержкой и отчёт о стабильности; документация по сетевой архитектуре и выбору модели синхронизации; презентация.

Сетевая архитектура и синхронизация — 35% (стабильность, корректность, устойчивость к задержкам); игровой дизайн и кооперативные механики — 25% (вовлечённость, баланс, необходимость командного взаимодействия); качество кода и масштабируемость — 25% (чистота архитектуры, возможность добавления новых радаров и сценариев); качество презентации и защиты — 15%.

№ 156. Кейс: Процедурная генерация космических явлений и ИИ-агенты для образовательной стратегии по солнечно-земной физике.

Институт солнечно-земной физики разрабатывает образовательную компьютерную игру в жанре стратегии, в которой игрок управляет исследовательским центром, изучающим влияние солнечной активности на Землю. В игре необходимо моделировать солнечные вспышки, корональные выбросы массы и их воздействие на магнитосферу, ионосферу и инфраструктуру Земли. Студенты выступают в роли разработчиков игровых систем.

Солнечная активность непредсказуема и многофакторна: каждая вспышка уникальна по интенсивности, направлению и составу. Предзаданные сценарии быстро надоедают игроку и не отражают реальную стохастическую природу явлений. Также необходимо, чтобы виртуальные «научные сотрудники» (ИИ-агенты) реалистично реагировали на события, принимали решения и взаимодействовали с игроком. Требуются система процедурной генерации и интеллектуальное поведение NPC.

Разработать систему процедурной генерации солнечных событий (вспышки классов А–Х, корональные выбросы, потоки солнечного ветра) с параметрами, основанными на реальных статистических распределениях. Реализовать ИИ-агентов (научных сотрудников обсер-

ватории), использующих деревья поведения (Behavior Trees) или конечные автоматы, которые анализируют сгенерированные события, выдают рекомендации игроку и реагируют на его команды. Интегрировать обе системы в игровой прототип.

Датасет реальных солнечных событий (GOES, NOAA) для калибровки генератора; описание физических параметров явлений; шаблоны деревьев поведения и FSM; игровой прототип с базовой архитектурой; геймдизайн-документ с описанием ролей NPC и игрового цикла; API игрового движка для работы с procedural generation и AI.

Модуль процедурной генерации с настраиваемыми параметрами распределений; система ИИ-агентов с визуализацией логики принятия решений; интегрированный прототип с демонстрацией взаимодействия агентов и генерируемых событий; отчет с описанием алгоритмов и обоснованием выбора архитектуры ИИ; презентация.

Качество процедурной генерации — 30% (разнообразие, реалистичность, параметризуемость); архитектура и поведение ИИ-агентов — 30% (логичность, адаптивность, читаемость кода); интеграция и работоспособность прототипа — 25% (отсутствие критических багов, согласованность систем); качество презентации и защиты — 15%.

№ 157. Написать генератор лабиринта.

2.3.17. Практические задания для оценки компетенции «ПК-8 ППК-Р7.1»

№ 158. Разработать игру Ping-Pong.

№ 159. Написать игру "Flappy Bird".

№ 160. Кейс: Разработка 2D-аркады «Перехват солнечного ветра» для популяризации гелиофизики.

Научный институт солнечно-земной физики готовит выставку для школьников и абитуриентов и заказывает небольшую 2D-игру для интерактивного киоска. Игра должна в увлекательной форме показать, как работает система мониторинга космической погоды: игрок управляет орбитальным спутником-детектором, который перехватывает частицы солнечного ветра и классифицирует их по типам. Студенты выступают в роли начинающих разработчиков игр.

На выставке посетителям сложно объяснить, чем занимаются учёные-гелиофизики, с помощью одних лишь плакатов. Нужна простая, но наглядная игра, которая за 2–3 минуты объяснит базовые понятия: солнечный ветер, типы частиц (протоны, электроны, альфа-частицы), роль спутников-мониторов. Игра должна быть лёгкой в освоении, работать на обычном ПК и не требовать сложной графики.

Разработать прототип 2D-аркады на любом игровом движке Godot Реализовать: управление спутником с клавиатуры или мыши, спавн частиц солнечного ветра с разными свойствами, систему классификации частиц по типам (захват нужным детектором), подсчёт очков и уровней сложности, простое меню и экран окончания игры.

Краткая справка о солнечном ветре и типах частиц (для игрового контекста); набор простых 2D-спрайтов (спутник, частицы трёх типов, фон звёздного неба); геймдизайн-документ с описанием механик и баланса; примеры похожих простых 2D-игр (Arkanoid, Space Invaders) как референс; базовые уроки по работе с 2D в выбранном движке.

Работающий прототип 2D-аркады, который можно запустить на обычном ПК; исходный код проекта с комментариями; краткое описание реализованных механик и логики игры; короткая презентация с демонстрацией геймплея и объяснением, как игра решает задачу популяризации науки.

Соответствие решению поставленной задаче — 40% (все заявленные механики работают, игра понятна); качество реализации и чистота кода — 25% (архитектура, отсутствие критических багов, читаемость); игровой баланс и UX — 20% (удобство управления, понятность правил, плавность геймплея); качество презентации и защиты — 15% (ясность объяснения решений и демонстрация).

№ 161. Кейс: Мультиплеерная кооперативная игра для совместного управления сетью радаров.

Институт располагает распределённой сетью из нескольких когерентных радаров, разнесённых географически. Для полноценного мониторинга ионосферы операторы радаров должны координировать свои действия в реальном времени. Институт заказывает разработку кооперативной мультиплеерной игры-тренажёра, где команда из 3–5 игроков совместно управляет виртуальной радарной сетью, реагируя на геомагнитные возмущения. Студенты — команда сетевых разработчиков.

Реальные операторы радаров работают удалённо и нуждаются в отработке навыков командного взаимодействия в стрессовых ситуациях (геомагнитные бури). Существующие тренажёры рассчитаны на одного пользователя и не моделируют сетевое взаимодействие. Разработка мультиплеера сопряжена с проблемами синхронизации состояния игры, задержек сети (latency), разрешения конфликтов при одновременных действиях игроков и обеспечения плавного игрового опыта.

Разработать прототип кооперативной мультиплеерной игры с сетевой архитектурой (клиент-сервер или peer-to-peer с авторитетным сервером). Реализовать: синхронизацию состояния радаров между игроками, систему ролей (каждый игрок управляет своим радаром), механику совместного покрытия зон наблюдения, механизм компенсации сетевых задержек (lag compensation / client-side prediction), систему голосовой или текстовой коммуникации между игроками, сценарии геомагнитных бурь с нарастающей сложностью.

Карта размещения реальных радаров с параметрами (дальность, угол обзора, частота); сценарии геомагнитных возмущений (таймлайны событий); библиотека сетевых фреймворков для выбранного движка (Mirror, Photon, Netcode for GameObjects и др.); требования к допустимой задержке (<100 мс); тестовые сценарии кооперативного геймплея; описание UX-паттернов для кооперативных интерфейсов.

Работающий сетевой прототип с подключением 3–5 игроков; демонстрация синхронизированного управления радаром в реальном времени; визуализация зон покрытия и их объединения; стресс-тест с искусственно добавленной задержкой и отчёт о стабильности; документация по сетевой архитектуре и выбору модели синхронизации; презентация.

Сетевая архитектура и синхронизация — 35% (стабильность, корректность, устойчивость к задержкам); игровой дизайн и кооперативные механики — 25% (вовлечённость, баланс, необходимость командного взаимодействия); качество кода и масштабируемость — 25% (чистота архитектуры, возможность добавления новых радаров и сценариев); качество

презентации и защиты — 15%.

№ 162. Кейс: Процедурная генерация космических явлений и ИИ-агенты для образовательной стратегии по солнечно-земной физике.

Институт солнечно-земной физики разрабатывает образовательную компьютерную игру в жанре стратегии, в которой игрок управляет исследовательским центром, изучающим влияние солнечной активности на Землю. В игре необходимо моделировать солнечные вспышки, корональные выбросы массы и их воздействие на магнитосферу, ионосферу и инфраструктуру Земли. Студенты выступают в роли разработчиков игровых систем.

Солнечная активность непредсказуема и многофакторна: каждая вспышка уникальна по интенсивности, направлению и составу. Предзаданные сценарии быстро надоедают игроку и не отражают реальную стохастическую природу явлений. Также необходимо, чтобы виртуальные «научные сотрудники» (ИИ-агенты) реалистично реагировали на события, принимали решения и взаимодействовали с игроком. Требуются система процедурной генерации и интеллектуальное поведение NPC.

Разработать систему процедурной генерации солнечных событий (вспышки классов А–Х, корональные выбросы, потоки солнечного ветра) с параметрами, основанными на реальных статистических распределениях. Реализовать ИИ-агентов (научных сотрудников обсерватории), использующих деревья поведения (Behavior Trees) или конечные автоматы, которые анализируют сгенерированные события, выдают рекомендации игроку и реагируют на его команды. Интегрировать обе системы в игровой прототип.

Датасет реальных солнечных событий (GOES, NOAA) для калибровки генератора; описание физических параметров явлений; шаблоны деревьев поведения и FSM; игровой прототип с базовой архитектурой; геймдизайн-документ с описанием ролей NPC и игрового цикла; API игрового движка для работы с procedural generation и AI.

Модуль процедурной генерации с настраиваемыми параметрами распределений; система ИИ-агентов с визуализацией логики принятия решений; интегрированный прототип с демонстрацией взаимодействия агентов и генерируемых событий; отчет с описанием алгоритмов и обоснованием выбора архитектуры ИИ; презентация.

Качество процедурной генерации — 30% (разнообразие, реалистичность, параметризуемость); архитектура и поведение ИИ-агентов — 30% (логичность, адаптивность, читаемость кода); интеграция и работоспособность прототипа — 25% (отсутствие критических багов, согласованность систем); качество презентации и защиты — 15%.

№ 163. Написать генератор лабиринта.

2.3.18. Практические задания для оценки компетенции «ПК-8 ППК-Р7.4»

№ 164. Разработать игру Ping-Pong.

№ 165. Написать игру "Flappy Bird".

№ 166. Кейс: Разработка 2D-аркады «Перехват солнечного ветра» для популяризации гелиофизики.

Научный институт солнечно-земной физики готовит выставку для школьников и абитуриентов и заказывает небольшую 2D-игру для интерактивного киоска. Игра должна в увлекательной форме показать, как работает система мониторинга космической погоды:

игрок управляет орбитальным спутником-детектором, который перехватывает частицы солнечного ветра и классифицирует их по типам. Студенты выступают в роли начинающих разработчиков игр.

На выставке посетителям сложно объяснить, чем занимаются учёные-гелиофизики, с помощью одних лишь плакатов. Нужна простая, но наглядная игра, которая за 2–3 минуты объяснит базовые понятия: солнечный ветер, типы частиц (протоны, электроны, альфа-частицы), роль спутников-мониторов. Игра должна быть лёгкой в освоении, работать на обычном ПК и не требовать сложной графики.

Разработать прототип 2D-аркады на любом игровом движке Godot Реализовать: управление спутником с клавиатуры или мыши, спавн частиц солнечного ветра с разными свойствами, систему классификации частиц по типам (захват нужным детектором), подсчёт очков и уровней сложности, простое меню и экран окончания игры.

Краткая справка о солнечном ветре и типах частиц (для игрового контекста); набор простых 2D-спрайтов (спутник, частицы трёх типов, фон звёздного неба); геймдизайн-документ с описанием механик и баланса; примеры похожих простых 2D-игр (Arkanoid, Space Invaders) как референс; базовые уроки по работе с 2D в выбранном движке.

Работающий прототип 2D-аркады, который можно запустить на обычном ПК; исходный код проекта с комментариями; краткое описание реализованных механик и логики игры; короткая презентация с демонстрацией геймплея и объяснением, как игра решает задачу популяризации науки.

Соответствие решению поставленной задаче — 40% (все заявленные механики работают, игра понятна); качество реализации и чистота кода — 25% (архитектура, отсутствие критических багов, читаемость); игровой баланс и UX — 20% (удобство управления, понятность правил, плавность геймплея); качество презентации и защиты — 15% (ясность объяснения решений и демонстрация).

№ 167. Написать генератор лабиринта.

3. Промежуточная аттестация

3.1. Методические материалы, определяющие процедуру оценивания знаний, умений, навыков и опыта деятельности

Экзамен является заключительным этапом процесса формирования компетенций обучающегося при изучении дисциплины и имеет целью проверку и оценку знаний обучающегося по теории, и применению полученных знаний, умений и навыков при решении практических задач.

Экзамен проводится по расписанию, сформированному учебно-методическим управлением, в сроки, предусмотренные календарным учебным графиком. Экзамен принимается преподавателем, ведущим лекционные занятия.

Экзамен проводится только при предъявлении обучающимся зачетной книжки и при условии выполнения всех контрольных мероприятий, предусмотренных учебным планом и рабочей программой дисциплины. Обучающимся на экзамене представляется право выбрать один из билетов. Время подготовки к ответу составляет 30 минут. По истечении установленного времени обучающийся должен ответить на вопросы экзаменационного билета.

Результаты экзамена оцениваются по четырехбалльной системе и заносятся в зачетно-экзаменационную ведомость и зачетную книжку. В зачетную книжку заносятся только положительные оценки. Подписанный преподавателем экземпляр ведомости сдается не позднее следующего дня в деканат.

В случае неявки обучающегося на экзамен в зачетно-экзаменационную ведомость делается отметка «не явка». Обучающиеся, не прошедшие промежуточную аттестацию по дисциплине, должны ликвидировать академическую задолженность в установленном локальными нормативными актами порядке.

3.2. Вопросы к экзамену

№	Вопрос	Код компетенции
1.	Что такое компьютерная игра. Жанры компьютерных игр.	ПК-2 ППК-Р2.1
2.	Графический движок Godot. Устройство и основные объекты. Возможности.	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
3.	Язык GDScript. Его применение.	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
4.	Обработка событий. Сигналы. Создание произвольных сигналов.	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
5.	Работы с графическими элементами. Загрузка изображений и рехмерных моделей. Проигрывание звука. Направление и эффект Допплера.	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3
6.	Вектора и операции над ними. Моделирование движения при помощи сил.	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
7.	Система частиц. Требования к системе частиц. Основные свойства частиц.	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
8.	Анимация. Виды анимации. Машина состояний.	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3
9.	Генерация карт. Применение двумерного шума Перлина.	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3

3.3. Тематика курсовых работ

По данной дисциплине выполнение курсовых проектов (работ) не предусматривается.

3.4. Материалы для компьютерного тестирования обучающихся

Общие критерии оценивания

Процент правильных ответов	Оценка
91% – 100%	5 (отлично)
81% – 90%	4 (хорошо)
71% – 80%	3 (удовлетворительно)
Менее 70%	2 (неудовлетворительно)

Соответствие вопросов теста индикаторам формируемых и оцениваемых компетенций

№ вопроса в тесте	Код индикатора компетенции
1	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
2	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
3	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
4	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
5	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
6	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
7	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
8	ПК-2 ППК-Р2.1
9	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
10	ПК-2 ППК-Р2.1
11	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
12	ПК-2 ППК-Р2.1
13	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
14	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
15	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
16	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
17	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
18	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
19	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
20	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
21	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
22	ПК-2 ППК-Р2.1
23	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3
24	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
25	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
26	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3
27	ПК-2 ППК-Р2.1
28	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3
29	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.3

Ключ ответов

№ вопроса в тесте	Номер ответа (или ответ, или соответствие)
1	b, d
2	b
3	a
4	a, b

№ вопроса в тесте	Номер ответа (или ответ, или соответствие)
5	a
6	a
7	a, b
8	a
9	a, d
10	c
11	b
12	b
13	a
14	b, d
15	b
16	c, d
17	b
18	a
19	d
20	a, c
21	a, b, c, d
22	c
23	b, d
24	c
25	b
26	c
27	b
28	b
29	b

Перечень тестовых вопросов

№ 1. Задание с множественным выбором. Выберите 2 правильных ответа.

Что такое автономный агент?

a. Сущность, действия которой основываются только на данных изначально заложенных при создании.

b. это интеллектуальный агент, который принимает свои решения о необходимых действиях без влияния из вне.

c. Сущность, действия которой направлены на достижение поставленной цели, не взаимодействуя с окружением.

d. Сущность, действия которой направлены на достижение поставленной цели, взаимодействуя с окружением используя сенсоры и актюаторы.

№ 2. Задание с единственным выбором. Выберите один правильный ответ.

Для чего используется ключевое слово puppet?

a. для обозначение дочернего класса

- b. для синхронизации состояний
- c. для создания удаленного клиента

№ 3. Задание с единственным выбором. Выберите один правильный ответ.

Для чего применяются аннотации в языке GDScript?

a. аннотации необходимы для внешних утилит и определяют то, как будет обработан скрипт

- b. аннотации используются для документирования кода
- c. аннотации меняют поведение метода или функции аналогично декоратору

№ 4. Задание с множественным выбором. Выберите 2 правильных ответа.

Какие игры можно разрабатывать на движке Godot?

- a. 3D
- b. 2D

№ 5. Задание с единственным выбором. Выберите один правильный ответ.

Что такое RPC?

- a. удаленный вызов процедур
- b. удаленное соединение игрока

№ 6. Задание с единственным выбором. Выберите один правильный ответ.

Какой метод используется для рисования объекта?

- a. `_draw`
- b. `_set_texture`
- c. `_process`

№ 7. Задание с множественным выбором. Выберите 2 правильных ответа.

Что такое компьютерная игра?

- a. Игра, построенная с использованием мультимедийных возможностей компьютера.
- b. Сложная программа, которая служит человеку для обучения или развлечения и рассчитана на разные возрастные группы.

№ 8. Задание с единственным выбором. Выберите один правильный ответ.

Верно ли утверждение, что язык GDScript является интерпретируемым языком программирования?

- a. Да
- b. Нет

№ 9. Задание с множественным выбором. Выберите 2 правильных ответа.

Какими свойствами можно настраивать генерацию шума на основе волн?

- a. частота
- b. сдвиг
- c. скорость
- d. амплитуда

№ 10. Задание с единственным выбором. Выберите один правильный ответ.

Каким образом изменится поведение частиц при установки параметра Local coords в disabled?

- a. Частицы будут испускаться из центра области и не будут выходить за нее

b. Частицы перестанут испускаться

c. Частицы будут испускаться из центра области и будут выходить за нее

№ 11. Задание с единичным выбором. Выберите один правильный ответ.

Каким образом можно сделать так, чтобы анимация не зависела от FPS?

a. ничего не нужно делать, так как анимация не зависит от количества выводимых кадров

b. необходимо учесть время между кадрами и рассчитать относительно него время между кадрами анимации

№ 12. Задание с единичным выбором. Выберите один правильный ответ.

Верно ли утверждение, что любой объект в Godot представлен в виде объекта типа Scene?

a. Нет

b. Да

№ 13. Задание с единичным выбором. Выберите один правильный ответ.

Какой параметр можно использовать для того, чтобы частицы вращались вокруг своей оси после испускания?

a. Angular velocity

b. Orbit velocity

c. Spin velocity

№ 14. Задание с множественным выбором. Выберите 2 правильных ответа.

Какое смещение в координатах необходимо учитывать при рисовании линий или прямоугольников с нечетной толщиной?

a. -1.

b. 0.5

c. 1.

d. -0.5

№ 15. Задание с единичным выбором. Выберите один правильный ответ.

Какой используется механизм для обработки возникающих событий?

a. слотов

b. сигналов

c. машина состояний

№ 16. Задание с множественным выбором. Выберите 2 правильных ответа.

Какое смещение в координатах необходимо учитывать при рисовании линий или прямоугольников с нечетной толщиной?

a. -1.

b. 1.

c. -0.5

d. 0.5

№ 17. Задание с единичным выбором. Выберите один правильный ответ.

Можно ли установить несколько обработчиков для одного сигнала?

a. Нет

b. Да

№ 18. Задание с единичным выбором. Выберите один правильный ответ.

Так как метод `_draw` кэшируется, то для рисования в зависимости от состояния необходимо использовать дополнительный метод. Какой?

- a. `queue_redraw`
- b. `paint`
- c. `draw`
- d. `_redraw`

№ 19. Задание с единичным выбором. Выберите один правильный ответ.

Какой ключевой объект используется для построения игр?

- a. `Context`
- b. `Node`
- c. `Engine`
- d. `Scene`

№ 20. Задание с множественным выбором. Выберите 2 правильных ответа.

Какие скриптовые языки можно использовать в Godot?

- a. GDScript
- b. Python
- c. C#

№ 21. Задание с множественным выбором. Выберите 4 правильных ответа.

Какие основные слои используются в автономных агентах?

- a. Слой действия
- b. Слой управления
- c. Слой движения
- d. Слой выбора

№ 22. Задание с единичным выбором. Выберите один правильный ответ.

Какой параметр отвечает за время жизни частицы?

- a. `decay`
- b. `health`
- c. `lifetime`
- d. `life`

№ 23. Задание с множественным выбором. Выберите 2 правильных ответа.

Какие форматы можно использовать чтобы загрузить трехмерную сцену в Godot?

- a. `.cad`
- b. `.obj`
- c. `.3ds`
- d. `.blend`

№ 24. Задание с единичным выбором. Выберите один правильный ответ.

В каких величинах измеряется громкость звука?

- a. См
- b. Гц

c. дБ

d. Кг

№ 25. Задание с единичным выбором. Выберите один правильный ответ.

Какой используется механизм для обработки возникающих событий?

a. слотов

b. сигналов

c. машина состояний

№ 26. Задание с единичным выбором. Выберите один правильный ответ.

Что такое keyframe?

a. начальный кадр анимации

b. конечный кадр анимации

c. каждый кадр в анимации остающийся неизменным

№ 27. Задание с единичным выбором. Выберите один правильный ответ.

Какой объект используется для создания звука идущего из определенной точки?

a. AudioBus

b. AudioStream

c. AudioChannel

d. AudioSpot

№ 28. Задание с единичным выбором. Выберите один правильный ответ.

На каком типе данных основаны сигналы?

a. InputEvent

b. Variant

c. Scene

d. Quaternion

№ 29. Задание с единичным выбором. Выберите один правильный ответ.

Может ли сигнал принимать параметры?

a. Нет

b. Да