2.2.1. Иерархические машины состояний (HSM): комплексное поведение NPC-моба

Блок: Блок 2.2. Программирование (Продвинутый) · **Полугодие:** 2 · **Класс:** 5-7 · **Продолжительность:** 40 мин

Тема: Иерархические машины состояний и комплексное поведение · **Инструменты:** ПРИМС (Cyberiada Graph Editor), Мастерская игр «Урсула»

📋 Паспорт урока

🎯 Цель	Познакомить учащихся с иерархическими машинами состояний (HSM) в ПРИМС: с несколькими родительскими контейнерами и дочерними состояниями. Научить проектировать комплексное реалистичное поведение NPC с приоритетами и сложными цепочками переходов, управляемыми переменной.
📚 Тип урока	Урок открытия новых знаний с практической работой в редакторе Кибериада GE.
👥 Формы работы	Фронтальная демонстрация, индивидуальный практикум за компьютерами.
🛠️ Оборудование	Проектов, компьютеры с доступом к редактор ПРИМС Кибериада GE и «Урсулу».

🎯 Планируемые результаты

Предметные

- Учащиеся понимают разницу между плоской и иерархической машиной состояний.

- Учащиеся понимают назначение иерархии в автоматах и умеют создавать структуру из двух суперсостояний и одного простого состояния.
- Учащиеся соблюдают правила маршрутизации в HSM: входящие стрелки направляют в конкретную стартовую дочернюю ноду, выходящие стрелки общей смены состояния тянут от границы родительского состояния.
- Учащиеся умеют использовать события изменения переменной для перехода между состояниями.

Метапредметные

Коммуникативные	• Умение описывать логику режимов сложного механизма на техническом языке состояний и переходов.
Регулятивные	• Декомпозиция сложного поведения на уровни. • Пошаговая отладка иерархии. • Умение упорядочивать сложную схему: аккуратное размещение дочерних блоков и выравнивание стрелок внутри родительской рамки.
Познавательные	• Моделирование реалистичного поведения агента: робот сначала физически добирается до места аварии и только потом начинает ремонт. • Анализ конфликтов между состояниями.

📚 Ход урока

Этап 1. Введение и актуализация (3 мин)

🖥️ Слайд 1: Титульный

Иллюстрация: сервисный робот-инженер едет на вызов к генератору для проведения ремонта.

📣 **Действия учителя:** Приветствие. Напоминание: в первом полугодии научились делать NPC с

состояниями и переменными. Сегодня — поднимаем на уровень: иерархия состояний и сложное поведение.

«В первом полугодии наш робот действовал просто: патрулировал по квадрату или чинил что-то прямо на месте, где застал сигнал. Но настоящий робот должен вести себя более реалистично:

- Если ему отдали команду чинить — он должен доехать до сломанного объекта и по прибытии достать инструменты и начать ремонт.
- Если отправили в патруль — он должен вернуться в свою патрульную зону.
- А если работу отменили — должен уметь вернуться в дежурное состояние.

Сегодня мы освоим профессиональный уровень ПРИМС — построим иерархическую машину состояний (HSM) с двумя суперсостояниями и научим робота переключать эти сложные режимы по сигналам от переменной».

Этап 2. Зачем нужна иерархия состояний (6 мин)

🖥️ **Слайд 2: Плоская vs иерархическая машина**

«В первом полугодии мы делали плоские машины состояний: все состояния на одном уровне. Когда состояний становится 5–7 — переходы запутываются, как спагетти. Иерархическая машина состояний (HSM) решает это: состояния вкладываются друг в друга».

Аспект	**Плоская машина**	**Иерархическая машина**
Структура	Все состояния на одном уровне	Родительские состояния содержат дочерние
Переходы	Между любыми состояниями	На уровне родителя (применяется ко всем дочерним)

Аспект	Плоская машина	Иерархическая машина
Приоритеты	Сложно выразить	Родительское состояние перекрывает дочерние
Пример	«Патруль», «Преследование», «Атака», «Отдых», «Бой» — все в ряд	«Спокойствие» → (Патруль, Отдых); «Тревога» → (Преследование, Бой)

> ℹ️ **Родительское состояние = режим**
>
> Родительское состояние — это режим, внутри которого есть несколько под-состояний. Пока активен родитель, активны и его дочерние состояния. Переход из родителя срабатывает для всех дочерних сразу — это удобно для «глобальных» событий (например, «нарушитель вторгся» → выход из спокойного режима, в каком бы состоянии внутри ни был NPC).

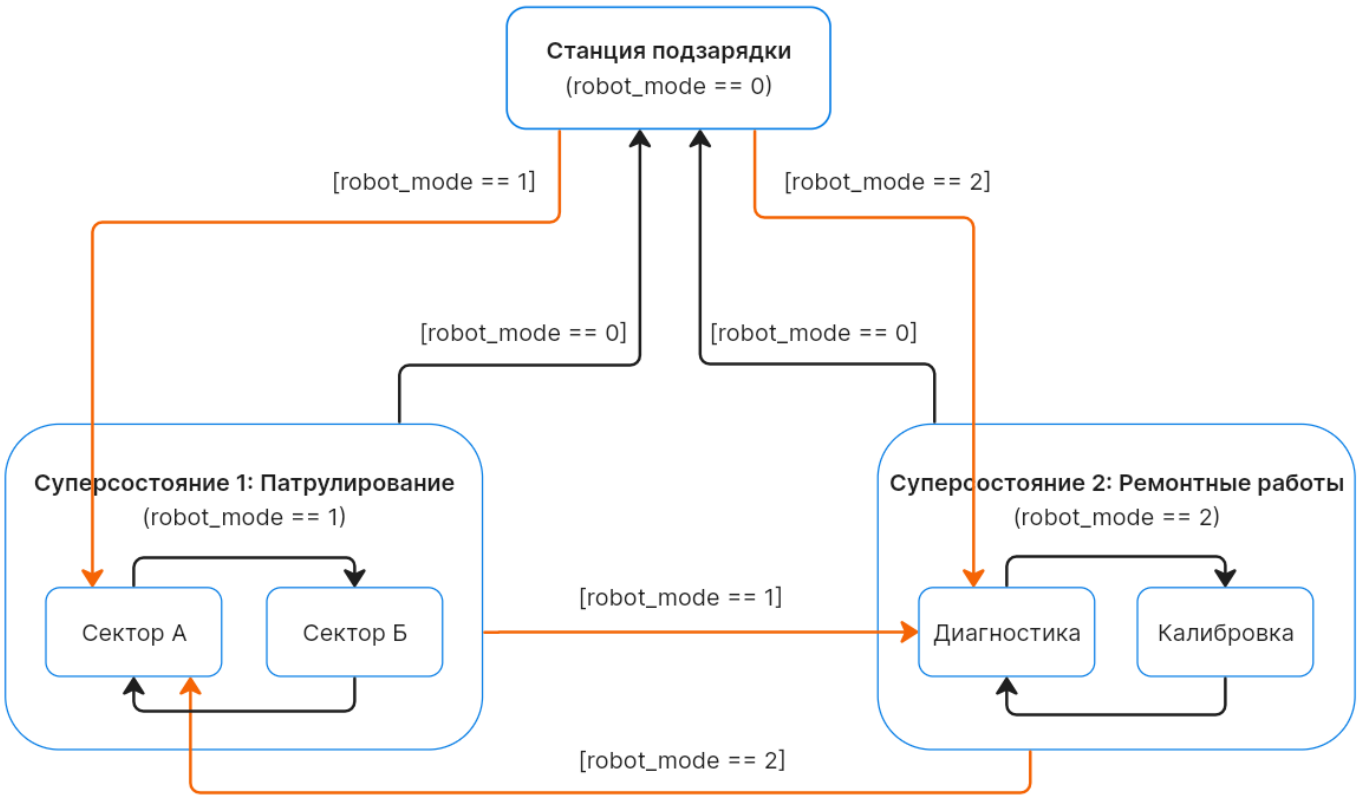
Этап 3. Демонстрация: NPC с тремя режимами (8 мин)

> 🖥️ **Слайд 3: Три режима сервисного робота**
>
> *Содержание слайда: Таблица и схема трех ключевых узлов: простое состояние «Станция подзарядки» и два родительских блока («Патрулирование» и «Ремонтные работы»).*

Учитель в реальном времени собирает HSM и комментирует:

Режим робота	Тип состояния	Что делает при входе	Внутренний цикл работы	Маяк
Станция подзарядки	Простое	Мониторит переменную. ВернутьсяНаИсходную(), проигрывает анимацию покоя	По внутреннему событию: когда прибыл на исходную — глушит моторы Стоп(),	Синий
Патрулирование	Суперсостояние	Мониторит переменную. ВернутьсяНаИсходную(), едет к зоне дозора	По дочерним состояниям: курсирует между Сектором А и Сектором Б	Зеленый

Ремонтные работы	Суперсостояние	Мониторит переменную. Находит объект для ремонта, едет к нему: ДвигатьсяКОбъекту()	Доехав, включает анимацию ремонта и переключается между дочерними состояниями Диагностика и Калибровка	Красный



📣 **Действия учителя:**

«При работе с иерархией действует строгое правило:

1. **Входить стрелкой нужно ВСЕГДА ВНУТРЬ — в конкретную первую дочернюю ноду.**

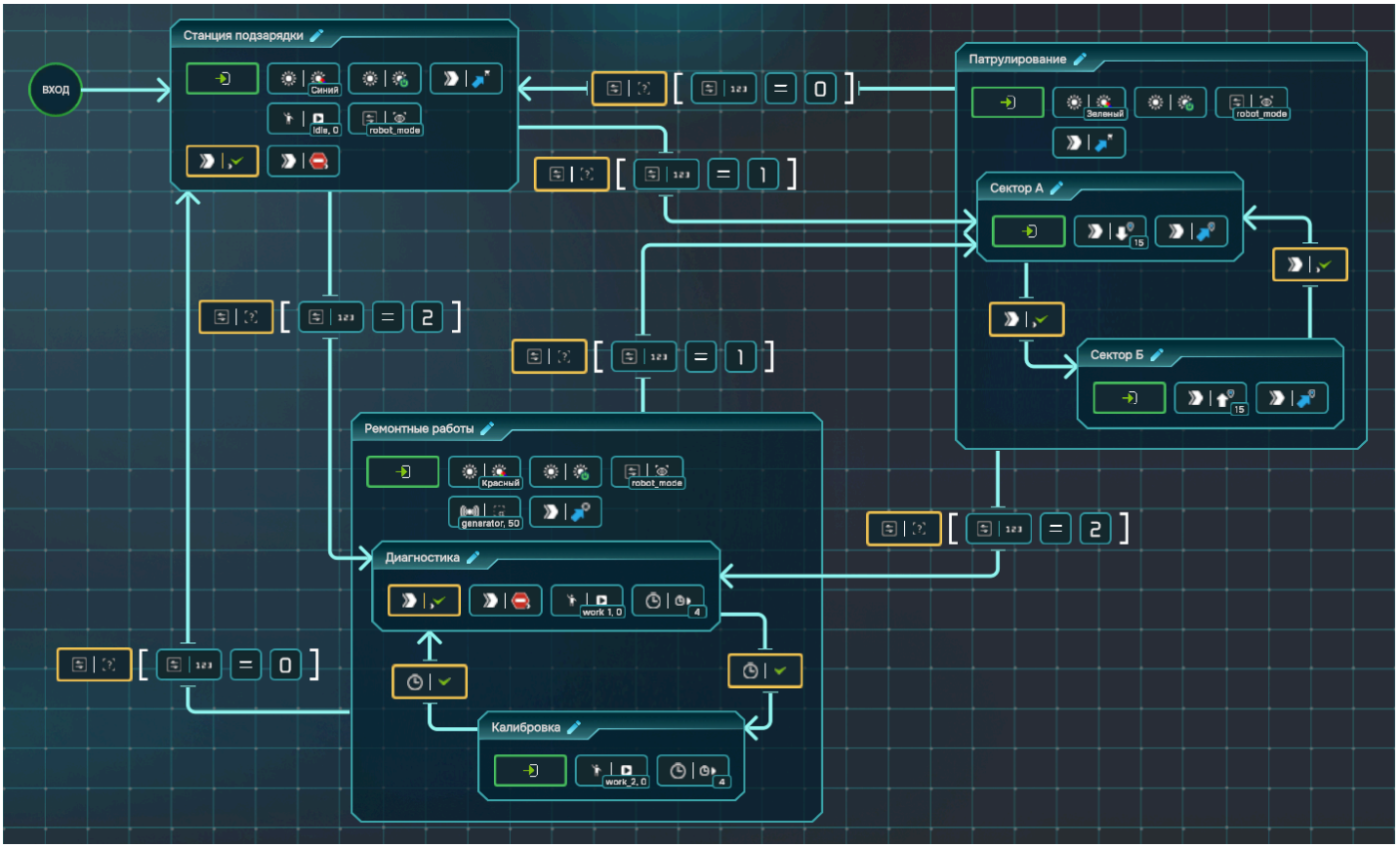
   Входя в режим патруля, стрелка целится в Сектор А. Входя в ремонт — в Диагностику. Сначала выполнятся общие команды родителя (включение маяка, поиск цели), а затем — команды первой точки.
2. **ВЫХОДИТЬ стрелкой смены режима нужно ОТ НАРУЖНОЙ ГРАНИЦЫ рамки.**

   В каком бы дочернем секторе или фазе ремонта робот ни находился, сигнал смены переменной снимет его с текущей задачи и переведет в новый режим целиком».

Этап 4. Разбор диаграммы робота (5 мин)

🖥️ Слайд 4: Демонстрация диаграммы в Кибериаде GE

Скриншот диаграммы robot_engineer.graphml с выделением внутренних циклов и связей между состояниями.



📣 Действия учителя:

«Посмотрите на режим ремонта: робот не воспроизводит анимацию ремонтных работ на ходу!

На уровне родителя Ремонтные работы он находит цель и просто едет к ней: ДвигатьсяКОбъекту().

А внутри, в первой дочерней ноде Диагностика, робот ждет события **ПеремещениеВыполнено**. Как только он доехал до объекта починки (генератора), робот глушит моторы (Стоп), запускает анимацию работы (work_1) и запускает наладку по таймеру на 4 секунды. Затем переключается на вторую анимацию работы в Калибровке.

Точно так при переходе в состояние Патрулирование: робот начинает ехать на исходную, и уже там патрулировать по циклу.

А сигналом к переключению режимов служит переменная robot_mode:

1 — патруль, 2 — ремонт, 0 — возврат в док подзарядки»

Прямо сейчас нам нужно завести ее в вашем игровом проекте, иначе при попытке собрать диаграмму в Кибериаде, мы не сможем работать с модулем Переменных. Конфиг редактора связан с конфигом игры, и, в том числе, в нем хранятся данные всех глобальных переменных.

Этап 5. Сборка диаграммы для робота-инженера (15 мин)

> 🖥️ **Слайд 5: Чек-лист сборки робота-инженера**
> *Содержание слайда:* Пошаговый план: ① Настройка станции подзарядки; ② Сборка суперсостояния патруля с возвратом в сектор патрулирования; ③ Сборка суперсостояния ремонта со сближением с объектом; ④ Соединение состояний стрелками переходов с условиями.

Каждый ученик настраивает диаграмму робота-инженера по шаблону. В дальнейшем он сможет применить этот шаблон к своим NPC.

Пошаговый план для учеников:

1. **Настраиваем станцию подзарядки:**
    - Создайте состояние Станция подзарядки (начальное от точки ВХОД).
    - На Входе: Индикатор(Синий), ВернутьсяНаИсходную(), Анимация(Idle, 0), СледитьЗаПеременной(robot_mode).
    - Добавьте внутреннее событие МодульДвижения.ПеремещениеВыполнено → команда МодульДвижения.Стоп().
2. **Собираем суперсостояние «Патрулирование»:**
    - Создайте родительский блок Патрулирование. На Входе: Индикатор(Зеленый), СледитьЗаПеременной(robot_mode), ВернутьсяНаИсходную().
    - Внутрь вложите дочерние ноды движения:
      Сектор А (вниз 15м, ДвигатьсяПоКоординатам);
      Сектор Б (вверх 15м, ДвигатьсяПоКоординатам).
    - Соедините их стрелками туда и обратно по событию ПеремещениеВыполнено.
3. **Собираем суперсостояние «Ремонтные работы»:**
    - Создайте родительский блок Ремонтные работы. На Входе: Индикатор(Красный),

СледитьЗаПеременной(robot_mode), ПоискОбъектаВРадиусеПоИмени(generator, 50), ДвигатьсяКОбъекту().

- Внутрь вложите дочернюю ноду Диагностика:

  Настройте внутреннее событие ПеремещениеВыполнено → команды МодульДвижения.Стоп(), ВоспроизвестиАнимацию(work_1, 0), ТаймерЗапуск(4).
- Также вложите внутрь дочернюю ноду Калибровка:

  На Входе → ВоспроизвестиАнимацию(work_2, 0), ТаймерЗапуск(4).
- Соедините Диагностику и Калибровку стрелками туда и обратно по событию Таймер.ТаймерВыполнен.

4. **Протягиваем стрелки смены режимов:**
   - От Станции подзарядки внутрь патруля к ноде Сектор А. По событию МодульПеременных.ЗначениеПеременнойИзменилось и по условию [robot_mode == 1].
   - От Станции подзарядки внутрь ремонта к ноде Диагностика. По событию МодульПеременных.ЗначениеПеременнойИзменилось и по условию [robot_mode == 2].
   - От границы рамки Патрулирование к Станции подзарядки. По событию МодульПеременных.ЗначениеПеременнойИзменилось и по условию [robot_mode == 0].
   - От границы рамки Патрулирование внутрь ремонта к Диагностике. По событию МодульПеременных.ЗначениеПеременнойИзменилось и по условию [robot_mode == 2].
   - От границы рамки Ремонтные работы к Станции подзарядки. По событию МодульПеременных.ЗначениеПеременнойИзменилось и по условию [robot_mode == 0].
   - От границы рамки Ремонтные работы внутрь патруля к Сектору А. По событию МодульПеременных.ЗначениеПеременнойИзменилось и по условию [robot_mode == 1].
5. Сохраните диаграмму как robot_engineer.graphml.

Этап 6. Подведение итогов (3 мин)

📣 **Действия учителя:**

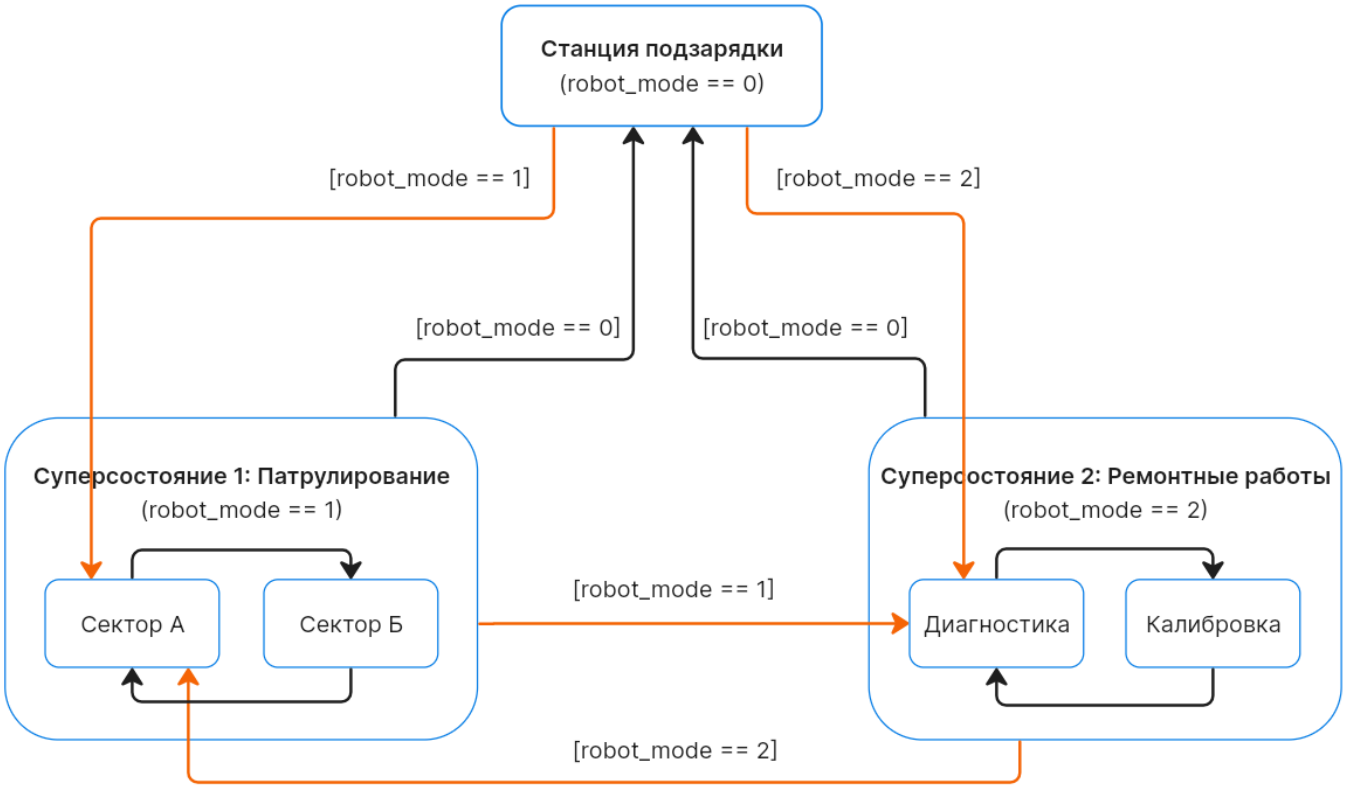
«Мы научили робота вести себя реалистично: он не размахивает инструментами во время перемещения, а сначала едет к объекту и начинает наладку только на месте. А если его отозвали в патруль — он возвращается в свой сектор и патрулирует там. На следующем уроке мы подключим к

этой схеме терминал управления с диалогом на Ink, чтобы протестировать переключение режимов и посмотреть, как иерархические машины состояний могут разнообразить механику игры».

> ℹ️ **Домашнее задание**
>
> Подумать, какие режимы есть в поведении вашего NPC (использовать библию персонажей из первого полугодия). Составить схему режимов одного NPC, содержащую не менее двух родительских состояний с несколькими дочерними внутри.

📄 Раздаточный материал

Схема конечного автомата робота-инженера



🎓 Методические рекомендации учителю

Урок вводит иерархию — это концептуально сложнее плоской машины. Главная идея: родитель = режим, дочерние = действия внутри режима. Покажите на демо, как переход «глобального» события мгновенно меняет весь режим NPC.

По итогам этого урока ученики еще не смогут протестировать робота-инженера, этому они научатся на следующем уроке. Поэтому стоит сосредоточиться на том, чтобы показать, как физическое перемещение к объекту отделяется от непосредственного выполнения работы. Ученики должны увидеть, что родитель отвечает за глобальное перемещение («едем к генератору»), а дочернее состояние — за локальную работу на месте («доехали → чиним (включили анимацию инструментов)»).

Предложите на этом уроке использовать базовую модель робота, которая есть в ассетах в «Урсуле». У нее есть удобные анимации починки (work_1 и work_2 — движения манипуляторами). Продемонстрируйте модель робота на проекторе — в игре или в окне просмотра 3D-моделей в любой программе.

💬 **Скрипт-помощник для работы с группами**

Ситуация	Фраза учителя
Ученик не понимает, зачем родитель	*«Родитель — это выключатель режима. Когда нужно сменить задачу на починку, ты не проверяешь, в «Секторе Б» или «на зарядке» был робот — ты просто включаешь режим «Ремонт». Все дочерние автоматически переключаются».*
Ученик жалуется, что стрелки запутались после создания родителя	*«Не переживай, это естественное поведение редактора при смене структуры. Отодвинь родительские рамки подальше друг от друга, выровняй блоки внутри по одной линии и потяни за стрелки, чтобы они шли прямо».*
Ученик хочет 3 уровня иерархии	*«Стоп. Два уровня — это уже сложно. Три — ты сам запутаешься. Оставь два, остальное — отдельными состояниями или отдельными NPC».*

⚠️ Возможные трудности и решения

Ситуация	Решение
Переход на уровне родителя не срабатывает	Проверьте: (1) событие-триггер указано; (2) условие (если есть) истинно; (3) цель перехода — родитель или его дочернее. В консоли отладки видно, какой переход пробуется.
NPC «зависает» в дочернем состоянии	Проверьте, есть ли выход из этого дочернего состояния. Если нет — добавьте переход (например, по таймеру или событию). Все дочерние должны иметь выход.
Слишком сложная иерархия (3+ уровня)	Для образовательного проекта достаточно 2 уровней (родитель + дочерние). Третий уровень — лишняя сложность. Если хочется — выносите в отдельный объект.

✅ Критерии оценки

✅ Успешно	❌ Требует доработки
• Созданы 2 родительских суперсостояния и 1 простое состояние подзарядки. • Внутри каждого суперсостояния настроен замкнутый цикл из двух дочерних нод. • Входящие стрелки ведут строго во внутренние стартовые блоки, выходящие — от границ рамок. • Все переходы между тремя основными режимами снабжены проверками переменной robot_mode.	• Схема плоская, родительские контейнеры не созданы. • Дочерние ноды обрываются в тупик, цикла нет. • Стрелка входа упирается в наружную границу родителя. • Переходы настроены без условий либо условия конфликтуют.

💡 Дополнительный совет

- Покажите демо на проекторе — иерархию трудно объяснить на словах, легко показать.

🏠 Домашнее задание

Подумать, какие режимы есть в поведении вашего NPC (использовать библию персонажей из первого полугодия). Составить схему режимов одного своего NPC, содержащую не менее двух родительских состояний с несколькими дочерними внутри.