

ХОЧУ В ГЕЙМДЕВ!



ОСНОВЫ ИГРОВОЙ РАЗРАБОТКИ
ДЛЯ НАЧИНАЮЩИХ

ВЯЧЕСЛАВ УТОЧКИН
КОНСТАНТИН САХНОВ

Вячеслав Уточкин, Константин Сахнов
Хочу в геймдев! Основы игровой
разработки для начинающих

© Уточкин В.Н., Сахнов К.С., текст, 2022

© ООО «Издательство «Эксмо», 2022

Введение

Создание качественной и успешной видеоигры – это комплексная работа, требующая от специалистов знаний и опыта в самых разных областях: программирования и гейм-дизайна, экономики и математики, истории и психологии, менеджмента, художественного и музыкального вкуса, знания маркетинга и продаж. Это далеко не полный список умений, необходимых для рождения вашей игры. Будет ли это просто и быстро? Нет. Есть ли универсальная формула успеха? Нет. Каждый игровой проект уникален и требует индивидуального решения задач, каждая игра – это свои вызовы и риски.

Нельзя научиться создавать игры, не создавая их. Именно ошибки, сделанные в ходе разработки собственной игры, позволяют получить опыт, определяющий вас как гейм-дизайнера. Пробуйте, ошибайтесь и пробуйте снова! Любая профессия, будь то повар, художник, актер или гейм-дизайнер, осваивается только с практикой. Создание игр – итеративный (циклический, повторяющийся) процесс: идея обдумывается снова и снова, превращается в многократно проверяемые прототипы, которые, в свою очередь, вырастают в бесконечно тестируемые версии, чтобы в итоге стать полноценным игровым продуктом. Чем больше тестов и проверок вы сделаете, тем больший опыт получите и тем быстрее придете к успеху.

Что-то обязательно пойдет не так, ведь сразу хорошо разобраться во всех нюансах игровой разработки не сможет ни один человек. Вам потребуются смелость и настойчивость, чтобы оттачивать навыки и принимать взвешенные решения, умение слушать не только себя, но и свою команду, своего инвестора и своих игроков – ведь, в конце концов, именно эмоции игроков определяют успех вашей игры.

Вне зависимости от того, какой старт выберет для себя человек – получение работы в игровой компании или путь инди-разработчика^[1], – везде важны профессионализм и качественный подход к делу. Каким бы сложным ни был процесс разработки игры, где все составляющие успеха еще и взаимосвязаны, все же нужно определить подзадачи и разобрать методы их решения. Каждый аспект достоин отдельной книги. Цель той, которую вы держите в руках, – предоставить начинающему создателю игр системные знания об игровом рынке в целом и прояснить методологию разработки игр как продукта.

«Эту книгу крайне желаю прочитать тем, кого зажигает игровая индустрия в качестве потенциальной карьеры и даже образа жизни. Игровая индустрия – Клондайк 21 века. Рекомендую».

МАКСИМ МИХЕЕНКО,

Co-founder/COO 5518 studios, инвестор в игровой индустрии

«Книга оставляет впечатление диалога с опытным наставником, который терпеливо рассказывает о разнообразных нюансах игровой разработки и отвечает на все вопросы».

ОЛЬГА МАКУШЕНКО,

Managing Director, PC & Console Games в 101XP.com

«Рекомендую к прочтению всем начинающим специалистам и тем, кто только планирует свой путь в игровой индустрии. Почему? Потому что она сэкономит вам много сил и нервов».

ГРИГОРИЙ ПОЛЯКОВ,

Продюсер, Geeky House

Люди и игры

Почему люди играют в игры

Сегодня игровая индустрия показывает стабильный рост, игровые продукты создаются при помощи новейших технических разработок, киберспортсмены в некоторых странах по популярности не уступают поп-звездам, а правительство США еще в далеком 2011 году признало видеоигры видом искусства.

Многие жанры видеоигр появились как возможность удовлетворить те или иные потребности человека. Зачастую новую игровую форму принимали явления, до этого уже существовавшие в физическом мире и выполнявшие те же функции. Рассмотрим некоторые из них.

Самая банальная причина любого хобби – это возможность приятно скоротать свободное время и удовлетворить некие психологические потребности, нереализованные в обычной жизни.

Первый вариант – просто убить время: поиграть в мобильную игру по дороге на учебу или провести вечер пятницы, расстреливая вражескую танковую дивизию. Главное – чтобы не было скучно.

Другой случай: люди уже целенаправленно выделяют время, чтобы поиграть. Обычно к такому досугу подходят более основательно: приглашают друзей, чтобы посвятить вечер настольным или кооперативным играм, перепройти в третий раз «Ведьмака», принимая другие решения, сходить в рейд в онлайн-игре, чтобы наконец прокачать персонажа до следующего уровня, и т. д. Важно уже не просто убить время, но получить определенные эмоции. И вряд ли выбор падет на небольшую мобильную игру с короткой сессией – скорее всего, это будет что-то более масштабное.

Подавляющее большинство игр дает пользователю ощущение награды за труды. В современном мире мы редко получаем награду сразу после того, как что-то сделаем. В древности, поймав условного мамонта, мы получали обед. Сегодня, работая с понедельника по

пятницу в офисе, мы видим награду пару раз в месяц – причем, скорее всего, она могла бы быть почаше и побольше. Деньги – отложенная и довольно условная награда. Даже прилагая усилия, мало кто из нас видит непосредственный результат своей деятельности, поэтому мы обращаемся за этим ощущением к хобби и развлечениям: ходим в походы, выращиваем цветы, рисуем открытки, качаем бицепсы или играем в игры.

Ощущение важности награды сильно варьируется в зависимости от аудитории игры. Кто-то еще радуется возможности украсить виртуальный дом, собирая шарики, или получить новую броню, убив монстра; для других такая награда больше не считается значимой и желанной. Видя результат собственных трудов, игроки все равно получают свою порцию дофамина и эндорфина (гормонов, отвечающих за положительные эмоции), но у большинства со временем притупляется желание «халявной» награды.

Сегодня игроки уже пресытились возможностью нажать на четыре кнопки и радоваться подарку за это. В играх все больше ценится вызов, когда мы сами ставим себе цели и достигаем их. Победив монстра в *Dark Souls*, мы уже не ждем огромной красивой надписи, восхваляющей наши таланты, – эмоции от того, что «это было сложно, но я все преодолел» гораздо сильнее. Хотя для китайских игр такие картинки во весь экран до сих пор остаются популярными.

За счет случайных, неожиданных поворотов игровых событий видеоигры могут обращаться к другой сильной эмоции – азарту. Азарт заставляет людей тратить огромные суммы в казино; ряд игр тоже делает ставку на предложение чего-то рискованного, непредполагаемого, делающего игровой процесс более захватывающим.

Следующая важная вещь – возможность получить особый опыт, отыгравая какую-то роль. Театр, маскарады, книги, даже детские игры в доктора и пр. позволяли кругу людей в определенных условиях переживать подобные эмоции. За счет таких жанров, как, например, RPG, возможность побыть кем-то другим (преступником,

героем-спасителем, ребенком) сегодня есть у каждого. Динамика игры может быть минимальной; старые RPG были пошаговыми или даже текстовыми. Игра может предлагать определенную роль или же давать возможность придумать ее самому в рамках игрового мира. За счет новой роли пользователя как главного действующего лица видеоигры позволяют еще глубже погрузиться в перипетии сюжета, чтобы ответственно принимать решения за своего персонажа.

Некоторые игры, напротив, обращаются к спинномозговым реакциям и действиям. Так играют, например, животные или маленькие дети. Мы получаем удовольствие от быстрой физической активности, дающей результат. Мы любим бегать, прыгать или кататься на лыжах; скоростные шутеры или сложные платформеры позволяют получать похожие эмоции, не выходя из собственной комнаты. Здесь большую роль играют скорость реакции и быстрое принятие решений, обеспечивающие победу или поражение.

Другое базовое удовольствие – получение новой информации. Мы путешествуем по миру, читаем научно-популярную литературу – другими словами, исследуем окружающий нас мир в погоне за этим ощущением. Исследование виртуальных миров позволяет ощутить схожие эмоции: любопытство, поиск ответов, радость получения новых знаний об игровом мире. Наверное, можно обозначить такой опыт как «виртуальный туризм».

Стратегии или головоломки, в свою очередь, дают возможность подумать, проверить собственный интеллект и умение планировать действия. Роли здесь чаще всего довольно условны, отыгрыша почти нет, реакция тоже отходит на второй план, главное – победа за счет умственных усилий.

Также игры дают возможность смотреть на нечто привлекательное. Например, красивая эльфийка с большими глазами привлекает мужскую аудиторию игроков. Естественно, не все игры делают ставку на нарочитую сексуальность, но количество фильмов для взрослых, героинями которых стали игровые

персонажи, говорит о том, что для части аудитории это довольно важно.

«Опять устался в свой компьютер, шел бы лучше на улицу с друзьями поиграл!» – переживают родители. Часто в представлении старшего поколения игры – это только карты, лото, конкурсы, настольные игры, то есть что-то, объединяющее людей, стимулирующее общение и совместный веселый досуг.

Когда компьютерные игры были уделом малочисленных гиков^[2], образ человека, предпочитающего одиноко сидеть перед мерцающим монитором вместо того, чтобы встретиться с друзьями, вызывал серьезные опасения и непонимание. Но этот период быстро закончился; за короткий срок компьютерные игры оценили миллиарды людей по всему миру, технологии шагнули вперед, и теперь видеоигры, как прежде их нецифровые предшественники, способствуют еще и общению, и социализации.

Желание быть лучшим, соревноваться или, напротив, кооперироваться и помогать друг другу наблюдается во всех видах деятельности человека, и видеоигры – отличная площадка для реализации таких потребностей. Первые будут стремиться обладать лучшими навыками и вооружением, чтобы легко «нагибать» любых соперников и быть в топах, вторые – часто с удовольствием играют, например, за хилеров^[3], получая благодарность других игроков. Многопользовательские игры дают возможность грамотно распределять роли и слаженно играть, чтобы успешно и интересно пройти предложенное задание.

Игровые паблики^[4] и игровые форумы создают новые площадки для общения по интересам. Плюс всевозможные системы кланов, союзов и так далее позволяют удовлетворить желание собирать вокруг себя группу единомышленников. Просмотр фильма или, например, чтение книг не могут дать подобных ощущений, главное преимущество видеоигр – их интерактивность. Вдобавок игры, в отличие от офлайн-активностей, позволяют общаться с людьми по всему миру.

В виртуальной реальности нет ограничений: здесь нет привязки к законам физики планеты Земля, государственных законов, необратимости смерти. Игры позволяют легко и безопасно создать ситуации, невозможные в реальности. Они реализуют всевозможные примеры компенсации упущенных или невозможных в обычной жизни событий и действий.

Если обобщить, мы можем сделать вывод, что люди играют в игры для получения опыта. Эмоционального, интеллектуального, а главное, уникального, собственного опыта. Почти нет игр, ориентированных только на одну потребность, обычно гейм-дизайнеры стараются передать некий спектр ощущений.

Опыт и эмоции абстрактны, но гейм-дизайнер должен уметь в них ориентироваться и знать, какими приемами и хитростями вызвать нужные ощущения у игрока. Вашей команде нужно четкое представление, над каким проектом вы вместе работаете, какие именно чувства и опыт должна дарить разрабатываемая игра. Только определив их, можно переходить к поиску конкретных игровых механик, визуального стиля и атмосферных деталей, с помощью которых вы попытаетесь этот опыт передать.

Как определить целевую аудиторию игры

Для кого мы делаем игры? Этот вопрос очень близко связан с целью создания игры вообще, и однозначного ответа он не имеет. Многие разработчики делают игры для себя, чтобы получить опыт или потому что им самим нравится конкретный жанр или сеттинг. Этот подход имеет право на существование, однако большинство гейм-дизайнеров все же хотят, чтобы с их детищем познакомилось как можно больше людей и, что немаловажно, чтобы игроки по достоинству его оценили.

Бывает, что вкусы гейм-дизайнера и аудитории не совпадают. К примеру, не все разработчики матч-3^[5] будут с удовольствием играть в них на досуге, но это не мешает создавать качественные и успешные игровые продукты. Необходимо научиться представлять себя на месте игроков, будь то тринадцатилетняя девочка из Китая или суровый небритый дальнбойщик из Челябинска. Чем больше вы будете знать о предпочтениях вашей аудитории, тем лучше сможете предугадать, какие особенности вашей игры могут принести им удовольствие. Более того, у вас будет понимание, где вообще эту аудиторию искать, через какие каналы продвигать продукт и так далее.

Все люди разные, и мы получаем удовольствие от разных вещей – в зависимости от наших предпочтений, возраста, пола, культуры и других многочисленных факторов. Сегодня конкуренция в индустрии более чем серьезная. В день выходит несколько тысяч игр, и какую бы замечательную игру вы ни создали, придется приложить усилия, чтобы о ней вообще хоть кто-то узнал.

Определять и понимать свою целевую аудиторию, ее желания, страхи, в том числе и скрытые в подсознании, предугадывать эмоции – всему этому предстоит научиться гейм-дизайнеру. Мы предлагаем следующее разделение игровых проектов по типу аудитории.

УСТОЯВШИЙСЯ МАССМАРКЕТ

Широкая аудитория, любящая популярные игры (*Call of Duty, PUBG, Overwatch, World of Warcraft* и др.). Потенциально такую игру может полюбить очень много людей, но разработчиков ждет жесточайшая конкуренция, поэтому подобные игры требуют максимального качества реализации, а также отточенного геймплея, уникальных особенностей, впечатляющей картинки. Чаще всего такие игры создаются узнаваемыми брендами с большим бюджетом для продвижения.

НИШЕВЫЕ ПРОЕКТЫ

Такие игры часто получают у людей, фанатеющих от своей идеи. Если у вас нет возможности вкладывать много ресурсов в изучение рынка, разных метрик (численных значений), маркетинговых стратегий и пр., можно делать вполне успешные игры для нишевой, то есть ограниченной аудитории. Чем больше окажется единомышленников, тем больше вы сможете заработать на таком нишевом проекте.

Например, можно делать игры для любителей классических RPG или космоса, хардкорных платформеров, поклонников определенной игры в духе старой школы и т. д. Если такую игру показать просто человеку с улицы, скорее всего, она ему не понравится. Например, выпускники программы «Менеджмент игровых проектов» в Центре развития компетенций в бизнес-информатике Высшей школы бизнеса НИУ ВШЭ сделали нишевую игру – симулятор лифта. Таких игр очень мало, и их проект стал популярным среди тех, кому интересны инди-игры с необычным геймплеем. Его скачали более полумиллиона игроков; это очень

хороший результат с учетом того, что разработчики не закупали рекламу.

Из-за специфики генерации трафика мобильные игры реже бывают нишевыми, в то время как на ПК (благодаря Steam и подобным сервисам) и – в меньшей степени – консолях игры для ограниченной аудитории могут быть довольно популярными.

ПРОЕКТЫ ДЛЯ УЖЕ НАБРАННОЙ АУДИТОРИИ

Такие игры создают не для широкой массы людей, а уже для своей аудитории или же аудитории другой конкретной игры. Если мы запускаем сиквел, то всегда можем рассчитывать на поклонников первой части. Или, если конкуренты переживают не лучшие времена, имеет смысл сделать игру для той же аудитории с любимыми механиками, но немного лучше. Можно сказать, что такие игры пытаются откусить свой кусок пирога от уже сформировавшегося, понятного сообщества, не уделяя много внимания привлечению новых пользователей. Хотя все равно следует помнить, что придется потратить определенные ресурсы на рекламную кампанию, чтобы донести до людей, почему они должны бросить одну игру и перейти в другую.

НОВОЕ СЛОВО В ИНДУСТРИИ

Самый рискованный вариант – это создавать аудиторию своего проекта с нуля. Если вы придумали новый игровой жанр или путем долгого поиска и прототипирования случайно наткнулись на что-то необычное и перспективное, можно предположить, что существует аудитория, которая оценит ваши идеи.

Так как это что-то свежее, прямых конкурентов у игры какое-то время не будет, и вы сможете претендовать на любовь и деньги всех

заинтересовавшихся. Предсказать размер такой аудитории очень сложно. Легендарным примером здесь является история *World of Tanks* от *Wargaming*, сначала безуспешно пытавшейся продать свою идею крупным студиям, которым игра казалась чем-то нудным и неинтересным. Сегодня у компании *Wargaming* огромная армия поклонников. Или же *Dota*, создававшаяся просто как модификация карты из *Warcraft III* и завоевавшая любовь игрового сообщества по всему миру. Принцип таких игр – «все или ничего».

На практике эти подходы часто совмещают. Бывает, что одна игра умудряется попасть почти под каждый из описанных выше типов аудитории. В качестве примера здесь можно привести *Death Stranding*. На момент выхода и первая часть «Ведьмака» была довольно оригинальной. Или *Need for Speed* – когда-то сделав что-то новое и оригинальное, разработчики просто продолжают снабжать свою аудиторию новыми частями любимой франшизы. Хотя неоднозначные отзывы на *World of Warplanes* или *Dragon Age 2* говорят о том, что даже крупные студии не могут предсказать результат попыток заигрывания одновременно и с нишевой – как правило, более хардкорной аудиторией, – и с новой широкой публикой.

В создании качественных игр без инновационных идей для устоявшейся аудитории нет ничего плохого, но многим разработчикам такая работа кажется недостаточно творческой.

Разрабатывая игру для себя, вы реализуете собственные идеи, но рискуете не найти понимания широких масс. Если вы – счастливчик, одновременно являющийся и гейм-дизайнером, и игроком, разделяющим убеждения широкой публики, а ваши идеи при этом еще никем не реализованы и конкуренции нет, это заявка на успех.

Пытаться угодить и понравиться категории игроков, к которой вы не относитесь, – довольно сложный и ненадежный путь. Гейм-дизайнер, конечно, должен представлять себя на месте игрока, но всегда есть риск не угадать. Даже опросы нельзя считать до конца

достоверными: люди редко понимают, чего на самом деле хотят, и еще реже умеют эти желания сформулировать.

Например, одна из важных задач на старте проекта – выбор сеттинга и визуального стиля. Маркетологи и продюсеры проводят исследование рынка, ключевых конкурентов и разумно предполагают, что можно спросить игроков, что же хотят они сами. Казалось бы, что проще – спроси у целевой аудитории, какой сеттинг они хотят видеть в вашей игре и какой должен быть арт-стиль, и выбери наиболее популярный. На практике все не так просто. Первой проблемой будет выделить ту самую целевую аудиторию и найти каналы доступа к ней. Чтобы ее сформулировать, нужно провести исследование конкурентов. Но допустим, вы справились. Что дальше?

Нужно составить опрос. Практика показывает, что, если вы попросите человека накидать ссылок на понравившийся ему арт, очень многие будут не готовы тратить на ваш опрос столько времени. Если же вы попросите ссылку на картинку, лучше всего отражающую сеттинг, во-первых, человек может выбрать ее не потому, что именно этот сеттинг он хочет, а потому, что там ему понравился арт. Далеко не все понимают, что такое сеттинг в играх. Во-вторых, если просить написать текстом название сеттинга, вы получите все что угодно: «Марвел», «Человек-паук», «хочу игру про драконов», «дайте принцесс», «Лавкрафт» и т. д. Ну и, наконец, в-третьих, готовы ли вы потратить время на ручную обработку 10 000 ответов?

Получается, авторы опроса все же вынуждены использовать заранее заготовленные варианты, из которых должен выбрать потенциальный игрок. А это значит, что вы можете как минимум не угадать и просто не включить в опрос самый популярный сеттинг, а как максимум – неверно сформулировать названия, не давая ни пояснений, ни примеров из других игр и артов.

Не пользоваться опросами – тоже не лучший вариант. Это хороший инструмент, и, как у любого инструмента, у него есть зона

применения. Да, можно спрашивать про визуальный стиль и сеттинг, в отличие от той же технической части. Но подходить к разработке таких опросников нужно крайне аккуратно. Ведь игроки зачастую не могут сформулировать, чего хотят: понравилась ли им механика, конкретный герой, его арт или элемент игровой вселенной.

Лучшим решением будет максимальная сегментация и очень точные описания. Статистика не терпит халатности, и допускать двусмысленности в прочтении никак нельзя.

Выбор подхода всегда сильно зависит от проекта, его целей и конкретной команды. По сути, это баланс трех вещей: стоимости (затрат на производство), потенциала проекта и рисков. Крупные компании могут позволить себе огромный штат сотрудников и лучшие технологии, чтобы одновременно работать над играми, которые точно принесут прибыль, и над экспериментальными проектами. Средние студии обычно просто делают надежные проекты, не гонясь за чем-то новаторским. Для инди-разработчиков такой выбор всегда болезненный: либо вы жертвуете потенциалом, заведомо не соревнуясь с титанами индустрии за долю внимания широкой аудитории и прибыль, либо сильно рискуете ничего не заработать.

Когда вы определитесь с базовыми принципами подбора своей аудитории, вам предстоит ее более глубокое изучение. Оно уже давно не ограничивается определением пола, страны и возраста, хотя и эти данные, безусловно, важны. Нишевые проекты больше сосредоточены на игровых предпочтениях: какой сеттинг или арт-стиль востребованы аудиторией, какие игры считаются среди этих людей эталоном. Начать следует с изучения имеющегося объема информации: тематические форумы и паблики, тренды поисковых систем, опубликованные топы игр и так далее.

Чтобы провести исследование, необходимо сначала сформулировать некую гипотезу, что требует творчества и аналитики. Методов проверки довольно много. Прежде всего – метод прямого опроса: например, предлагаем пользователю

выбрать из нескольких картинок, с его точки зрения, лучшую. Или А/Б-тестирование: одной группе показываем одно, другой – другое, и смотрим на общий результат – допустим, кто больше нажал на кнопку (количество групп пользователей здесь может быть любым).

Крупные компании могут позволить себе отслеживать физические показатели участников фокус-групп (учащение пульса, направления взгляда) и на их основании делать выводы о привлекательности чего-либо.

Самая сложная часть – правильно сформулировать гипотезу. Мало кто рискует начинать это дело с нуля, гораздо проще предложить какое-то расширение механик для уже понятной аудитории. Допустим, мы знаем, что есть любители игр про постройку загородных домов. Можно предположить, что, если мы введем механику разграбления этих домов, игра станет интереснее. Здесь можно уже проводить опросы, описывая игровую ситуацию и предлагая варианты, например, хотят ли игроки улучшить охрану дома, поговорить с грабителем и заставить его раскаяться или, наоборот, расстрелять его из дробовика.

Психология игрока

Игроков много, и все они разные. Важно учитывать степень знакомства с жанром, платформой, конкретно вашим проектом. Кто-то играет каждый день по полчаса, другие выделяют время только раз в неделю, зато предаются любимому занятию 6–8 часов. Одни будут играть только с друзьями, другие готовы к командной работе со случайными людьми. Кто-то готов платить за игру регулярно, а кто-то, потратившись один раз на возможность играть, будет раздражен предложениями купить еще что-то внутри игры.

Делая предположения о том, что может понравиться вашей аудитории, следует помнить, что она неоднородна. И очень часто придется делать выбор в пользу того или иного решения, приятного для одной группы людей и неприемлемого для другой. Обычно гейм-дизайнер работает с конкретной категорией игроков, ведь создать игровой элемент, одинаково популярный для всех, – почти невыполнимая задача.

Экспериментальные игровые проекты, претендующие на привнесение в индустрию чего-то нового, сосредоточены на том, чтобы отыскать какой-то нереализованный запрос игроков. Например, создатели первых battle royale-игр четко уловили как интерес молодой аудитории к подобному жанру, так и общий тренд на соревновательные игры. Интерес к соревнованию на выживание идет из популярной литературы и кино начала 2000-х. Это и японский роман «Королевская битва», и крайне популярная среди молодежи фантастика вроде «Дивергента», «Бегущего в лабиринте», «Голодных игр». К слову, режим с таким названием появился в виде мода к *Minecraft* еще в 2012 году. Создатели современных battle royale подметили, что в то время ценность жизни в играх резко падала. Геймеры стали считать игры какими-то рафинированными, казуальными, где ты и не проигрываешь, и не выигрываешь. Пропало ощущение, что игра может серьезно «наказать» за

неудачные решения. На волне этих настроений радость от победы над реальными 99 соперниками ценилась в миллион раз больше, и королевские битвы на время стали одним из самых популярных жанров среди action-игр.

Есть классические исследования, выявляющие, какой вид удовольствия ищут разные группы людей, играя в игры. Конечно, они все не идеальны и полной картины, к сожалению, не дают. Это просто попытки объяснить в научной форме сложнейшие мозговые процессы, происходящие в реальности. Но для нас эти исследования все равно остаются полезными, так как помогают выделить основные паттерны поведения и сегментировать аудиторию.

Пожалуй, самой известной классификацией игроков по предпочитаемому ими типу удовольствия является классификация гейм-дизайнера и профессора Университета Эссекса Ричарда Бартла.

Бартл старался понять «плоскость интересов» разных групп людей в зависимости от того, что для них важнее – собственные действия или взаимодействие с другими людьми, игровой мир сам по себе или действия в нем. Он делит людей, играющих в многопользовательские игры, на четыре группы: ачиверы^[6], киллеры^[7], социальщики и исследователи. Многие законы взаимодействия, подмеченные Бартлом, можно применять и для однопользовательских игр.

Главное удовольствие **АЧИВЕРОВ** (еще их называют «карьеристами») – это вызов. Они выполняют различные игровые цели в погоне за радостью достижения определенных результатов, то есть для них важны именно выполняемые игровые действия и следующие за ними достижения. Обычно такие игроки стремятся к собирательству различных ресурсов, умений, артефактов, наград. В целом им важен внутриигровой прогресс как возможность достичь успеха. Практически все игровые платформы имеют систему достижений, и многие игроки с радостью тратят время на игру, чтобы получить коллекционную карточку или редкий значок.

КИЛЛЕРАМ нужна победа. Этот тип игроков ищет состязаний, прежде всего с другими людьми, чтобы показать, что они лучшие. Главное для киллеров – так или иначе сопоставлять себя с другими игроками, чувствовать свою значимость и превосходство, причем добиваться этого быстро – рутины киллеры не любят.

ИССЛЕДОВАТЕЛИ предпочитают взаимодействие с игровым миром, стремясь открыть его во всей полноте. Сражения и активные действия для них отходят на второй план. Исследователи ценят многообразие игрового контента, сюжет, разные игровые механики, позволяющие досконально изучить возможности игры; рейтинги и соревнования им неинтересны.

СОЦИАЛЬЩИКИ больше всего ценят взаимоотношения с другими игроками, чувство товарищества и собственную популярность. Часто такие люди, находя что-то интересное в вашей игре, приводят своих друзей и знакомых. Именно они чаще и активнее всего пользуются внутриигровым чатом, участвуют в жизни форумов, оставляют отзывы, становятся лидерами мнений.

На самом деле, естественно, не все люди хотят только описанных выше эмоций. Бартл занимался многопользовательскими играми, поэтому ко многим игровым жанрам (например, гонки) эта теория подходит с натяжкой. Позже Бартл усложнил системы деления на психотипы, добавив критерий «осознанности-неосознанности». Но на практике разработчики очень редко этим пользуются из-за невозможности объективной проверки, насколько то или иное действие игрока осознанно.

Как и в случае с другими классификациями, деление весьма условное. В одном человеке могут сочетаться несколько классов (а то и все, но какие-то будут доминирующими).



Рис. 1. Четыре психотипа игроков (типология Ричарда Бартла)

По четырем психотипам Бартла собрано много полезной информации (метрики по возвращению в игру, процент платящих игроков и пр.), поэтому для определения своей аудитории гейм-дизайнеры зачастую пользуются именно этой упрощенной системой.

Идеального деления на психотипы не существует. Для массовых многоплановых игр с простыми игровыми активностями (например, MMORPG) деление по Бартлу подходит очень хорошо. Цель таких игр – развлечь всех. Учесть все возможные желания и нужды очень сложно, поэтому допустимо мыслить широкими категориями. Необходимо видеть тренды, анализировать взаимодействие разных игроков, чтобы адекватно построить и поддерживать игровой процесс. Если же вы работаете над созданием гиперреалистичного авиасимулятора, где, чтобы поднять вертолет в воздух, нужно прочитать тридцать страниц инструкции, ваша главная задача – передать игрокам реалистичность и сложность процессов. Такую

игру можно не подстраивать под все психотипы, потому что сразу понятно, что это игра для «исследователей».

Игроки могут проявлять склонность к разным паттернам поведения, взаимодействуя с разными играми или даже с отдельными фидами внутри одной игры. В частности, играя в большие MMORPG старой школы, такие как *Perfect World*, игрок может проявлять себя представителем тех психотипов, которые ему наиболее близки. Все дело в том, что игра дает возможности для раскрытия каждого психотипа. В то же время игры типа «*Цивилизации*», «*Ведьмака*» или *Mass Effect* являются более узконаправленными, чем многообразные по своей природе MMORPG. Такие игры не позволяют каждому игроку раскрыть все свои психотипы. Если он, допустим, на 50 % киллер и на 50 % ачивер, то кем он проявит себя в «*Ведьмаке*»? Больше ачивером? Именно поэтому сам Бартл писал, что его модель разработана в первую очередь для открытых виртуальных миров, где у игрока есть возможность проявить себя в любой ипостаси.

Изменения вашей игры воздействуют на все группы, и вам как гейм-дизайнеру важно уметь предсказывать, как это нововведение повлияет на соотношение условных киллеров, ачиверов и пр. Суметь построить баланс активностей, интересных всем нужным вам типам игроков, – непростая задача.

BRAINHEX

Геймдев, как и другие отрасли, опирающиеся на маркетинг, пытается постоянно совершенствовать методы изучения своей аудитории. Крупные торговые компании выяснили, что определенные ароматы или цвета способствуют увеличению продаж. В экономике появились целые направления – нейроэкономика и нейромаркетинг, изучающие влияние различных факторов на наш мозг с целью повлиять на пользователя.

После долгих лет исследований получилось найти связи между гейм-дизайном и удовлетворенностью игрока. Так появилась модель BrainHEX, которой пользуются игровые студии по всему миру.

Ученые изучают мозговые центры, работающие во время игры и отвечающие за разные эмоции. Новые исследования ведутся постоянно, информация есть в открытом доступе, поэтому мы не будем задерживаться на биологической стороне вопроса. Прежде всего нас интересуют определенные «нейромедиаторы»^[8]:

- эндорфин, отвечающий за мобилизацию, нечувствительность к боли, концентрацию;
- адреналин (бодрость, напряжение, беспокойство);
- дофамин (любопытство, удовольствие);
- окситоцин (помогает относиться к чему-то с доверием, нежностью и любовью, уменьшает тревогу);
- норадреналин (напряжение, страх, стресс, шок).

Играя, человек получает эмоции именно за счет комбинации определенных гормонов. Они определяют, что мы чувствуем в данный момент: любопытство, вызов или желание разбить телефон о стену. Понимая, что происходит в голове у игрока, разработчики игр могут предположить реакцию на то или иное игровое событие и придумать способ монетизировать его. Естественно, на одну и ту же фишку люди могут реагировать по-разному. Все это сложные мозговые процессы, но на их основе удалось создать теорию о разделении игроков на группы.



Рис. 2. Модель BrainHEX

1. CONQUEROR (ЗАВОЕВАТЕЛЬ)

Чем-то этот типаж похож на бартловского киллера, но акценты немного смещены: легкая победа не интересует таких игроков. Сталкиваясь с препятствиями и ожесточенно сражаясь, мы становимся агрессивными и замотивированными на успех. Адреналин и норадреналин вызывают чувство раздражения, гнева. Для многих людей они являются источником мотивации, и победа, добытая потом и кровью, становится более значимой и приносит реальное удовлетворение.

Такие игры, как *StarCraft*, *Nioh* или *Dark Souls/Sekiro: Shadows Die Twice*, позволяют ощутить себя триумфатором, который смог добиться успеха, несмотря на ярость, сложность и отчаяние. В реальной жизни такие эмоции доступны крайне редко, поэтому «завоевателей» довольно много.

2. ACHIEVER (КАРЬЕРИСТ / ДОСТИГАЮЩИЙ ЦЕЛЕЙ)

Если на первый план для вас выходит удовольствие от того, что задание выполнено, то вы – коллекционер. *World of Warcraft, Monster Hunter*, баттлеры и другие подобные игры позволяют получить дофамин за счет той или иной награды и достижения результатов.

3. SURVIVOR (ВЫЖИВШИЙ)

Когда мы испытываем страх или стресс, в организме вырабатывается адреналин. Многие люди ненавидят подобные ощущения, но есть категория игроков, реально наслаждающихся острыми эмоциями и сильным волнением – чем сложнее, тем лучше. Ходить по краю, спастись от ужасной угрозы, а затем снова почувствовать себя в безопасности – такие эмоции позволяют ощутить, например, *Resident Evil* или *F.E.A.R.*

4. DAREDEVIL (БЕЗРАССУДНЫЙ, СОРВИГОЛОВА)

Рискованные действия, волнение от погони, возбуждение также вызывают всплеск адреналина. Это не просто наблюдение со стороны за скоростными перемещениями, игрок получает острые ощущения от риска, успешного прохождения сложных ситуаций и принятия быстрых решений. *Grand Theft Auto, Call of Duty*, сложные платформеры заставляют таких игроков поволноваться и получить удовольствие.

5. SEEKER (ИСКАТЕЛЬ)

Сталкиваясь с чем-то интересным и любопытным, наш мозг вырабатывает эндорфин. Исследования и открытия, устойчивый интерес ко всему новому, желание разобраться – все это можно получить, играя, например, в *The Elder Scrolls*, *The Legend of Zelda* или *Final Fantasy*. В той или иной степени исследования добавляются почти во все игровые жанры – другой вопрос, что не всегда это делается для монетизации игроков.

6. MASTERMIND (МЫСЛИТЕЛЬ, УМНИК)

Когда мы продумываем и реализуем стратегию для решения сложной задачи, мозг вырабатывает дофамин. Для любителей *Civilization*, *Myst* и т. д. процесс принятия этих решений, логика рассуждений становятся ключевыми факторами на пути к победе.

7. SOCIALIZER (СОЦИОФИЛ)

Главное удовольствие таким игрокам приносит окситоцин, отвечающий за чувство единения с другими людьми. Социофилы стремятся создать вокруг себя атмосферу доверия и обижаются, если кто-то нарушает ее. В контексте игры именно совместная активность определяет историю и впечатления, как, например, в *World of Warcraft* или *Second Life*.

Помните, что люди в реальности и в игре могут вести себя по-разному. В большинстве случаев в одном человеке спокойно уживаются 2–3 ведущих игровых психотипа, и не стоит по ним делать выводы о характере реального человека.

Теории о сегментации игроков – это инструмент для разработчиков, позволяющий оценить нововведения на востребованность для аудитории, выявить ее потребности, но, как и всегда, теория несколько упрощает реальность. Стремление дарить

только положительные эмоции нередко ведет к провалу: в играх ценится именно переход от одного ощущения к другому, сложность должна сменяться минутами отдыха, напряжение – вознаграждением за труды. Любое ощущение вызывает определенное привыкание, игроку становится скучно, поэтому именно разнообразие делает вашу игру привлекательной. Разработчики игр называют такую гармонию состоянием потока (flow).

Большая конкуренция заставляет игровые компании пользоваться результатами таких исследований, чтобы не только продать игру, но и реально выяснить интересы и ожидания игроков. Просто делать хорошие игры, не опираясь на маркетинг, знание своей аудитории и понимание стоимости трафика, – наивно, особенно если вы рассчитываете на финансирование своего проекта.

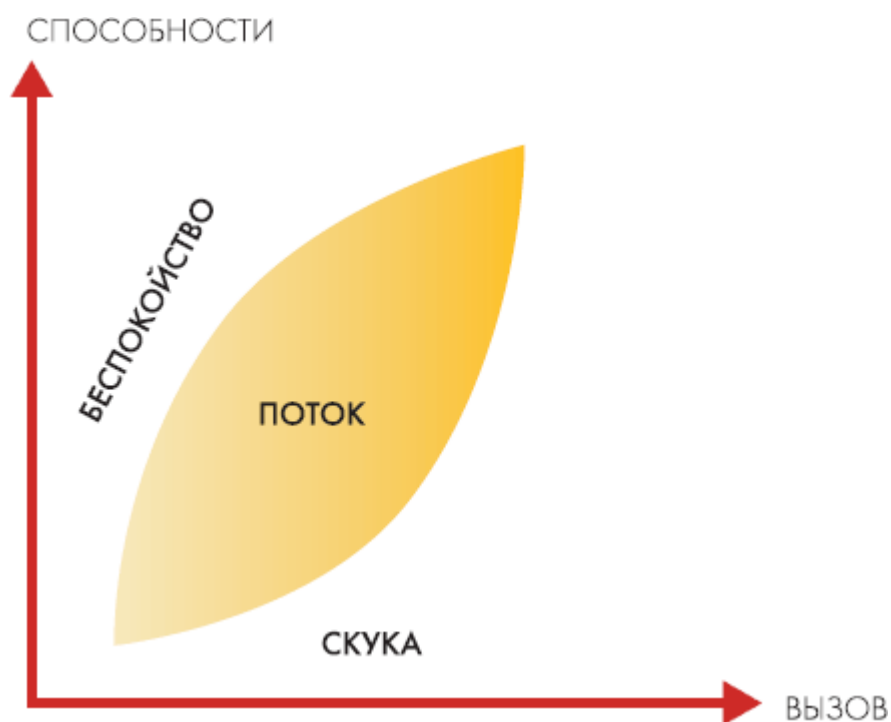


Рис. 3. Состояние потока

Изучение природы человеческих реакций и получения опыта продолжает развиваться в рамках игровой индустрии: UX^[9], психология игрока, аналитика и другие исследования рынка помогают продать игры и, что не менее важно, понять, какие ощущения делают людей счастливыми. Однако, пока игра не доделана, все предположения об аудитории могут оказаться неверными, особенно если у разработчиков пока мало опыта.

Игры и риски

Игры – это венчурный, то есть связанный с риском, бизнес. Даже подробный анализ игрового рынка за несколько лет в динамике не дает гарантий, что к моменту выхода игры предпочтения аудитории и технологии не изменятся.

Риски крупных компаний исчисляются сотнями миллионов долларов; у инди – несравнимо меньшими цифрами, а значит, такие команды могут преуспеть там, куда крупные компании просто не станут соваться: игры нишевые, странные, экспериментальные. Иногда лучшим решением для инди бывает отстраниться от трендов, стараясь выделиться за счет оригинального продукта.

Многие взрослые игроки тоскуют сегодня об «индустрии, которую мы потеряли». Они считают, что игровые компании думают только о показателях, зарплате и лутбоксах^[10] вместо того, чтобы сосредоточиться на свежих механиках, реально проработанном сюжете и атмосфере, впечатления от которых навсегда останутся в памяти (хотя второй раз, скорее всего, те же приемы и не сработают).

Открывая различные топы игр, мы часто видим бесконечные сиквелы и приквелы, а не только оригинальные проекты. Но на это есть объективные причины. Прежде всего, цена создания хорошей игры сильно выросла. Студии не готовы так рисковать: куда безопаснее выпустить очередную часть любимой франшизы, чем надеяться, что экспериментальная игра когда-нибудь окупится. Хотя на самом деле и здесь есть ряд великих проектов крупных студий, взять тот же *Overwatch* от *Blizzard*.

Вторая причина уже в восприятии. Если человек рос в то время, когда все выходящие игры были разными, каждый проект вообще не был похож на предыдущий, то вполне естественно, что сегодня такому игроку не хватает нового опыта. Но ту же тенденцию мы видим в кино и музыке – действительно, удивляют нас крайне редко,

при этом такие неэкспериментальные проекты имеют коммерческий успех. Более молодые люди не так критически относятся к клонам или новому прочтению давно знакомых произведений.

Есть мнение, что титаны индустрии, ориентируясь на самую широкую аудиторию, выпускают игры, которые будут приятны всем, но не любимы никем. Такие игры редко открывают новые горизонты, меняют чью-то жизнь, заставляют задуматься. Зато они выполнены качественно, они красивые и прибыльные, в них с удовольствием играют все.

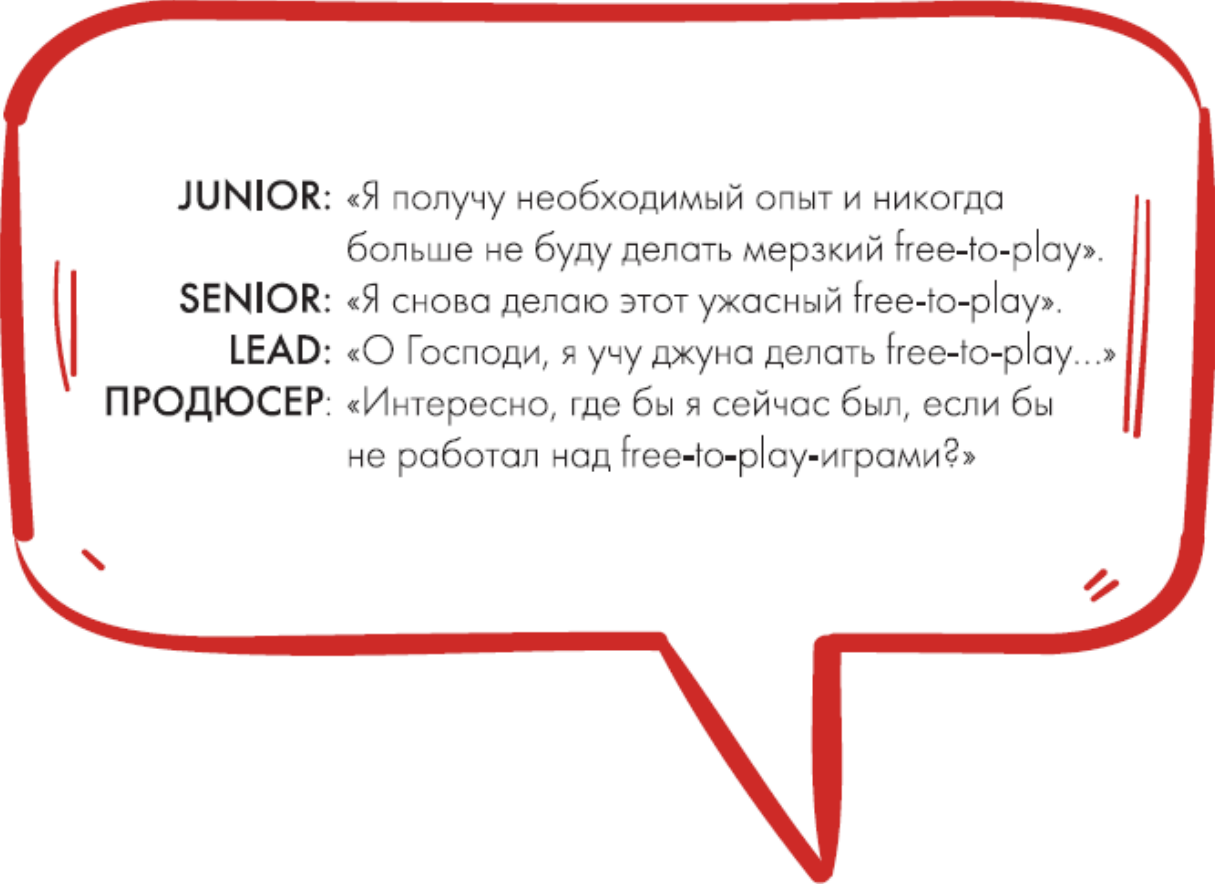
Ситуация улучшается за счет инди-студий; игровые движки, появление процедурной генерации и средств цифровой дистрибуции позволяют всем желающим начать разрабатывать игры. Если рассматривать инди не с точки зрения того, что они уступают по качеству играм массмаркета, а попробовать сравнить их, например, с арт-хаусным кино, получится, что именно инди могут себе позволить ориентироваться на собственные идеи, а не на исследования рынка, так как их риски меньше. Крупные игровые компании это понимают и постоянно ищут эти небольшие студии и их интересные проекты, чтобы приобрести для расширения собственного каталога игр.

По-своему игры Кодзимы^[11] – тоже инди, просто с другими бюджетами и аппетитами. Его творения ждут и любят миллионы именно за то, что они «не для всех», за уникальные механики и непривычные, новые идеи. Так что зачастую лучший вариант для инди-команд – креатив и заведомая «нишевость» проекта.

Если у вас мало опыта, возможно, для начала имеет смысл сделать небольшую игру для ограниченной аудитории, зато с оригинальной идеей или необычным геймплеем; получить навык и опыт, после чего ваши шансы быть замеченным крупной студией или инвестором многократно возрастут. На базе полученных знаний вы сможете либо «раскачать» существующий проект, либо сделать с помощью этих наработок более масштабную игру.

Такой опыт можно получить на Junior-позициях в игровых компаниях, и это тоже решение. Но если по каким-то причинам игра, над которой вы будете там работать, вам не нравится, спросите себя, стоит ли оно того. Сможете ли вы разрабатывать 128-й сценарий роста кукурузы для фермы, хотя всегда мечтали делать шутеры, или же вы просто решите, что игры – «не ваше», и пойдете искать себя в чем-то еще? Любовь к своему проекту – это действительно необходимая и лучшая мотивация.

Да, только получив рабочие прототипы и «играбельную» версию, вы сможете с уверенностью сказать, что ваша команда вообще способна выполнить проект в желаемом качестве. Только soft launch (запуск проекта для ограниченной аудитории с целью сбора информации) с удовлетворительными метриками покажет, что ваши продуктовые идеи работают. Но если ваша игра вам нравится, то шансы завершить разработку и найти пусть нишевую, но свою аудиторию намного выше.



JUNIOR: «Я получу необходимый опыт и никогда больше не буду делать мерзкий free-to-play».

SENIOR: «Я снова делаю этот ужасный free-to-play».

LEAD: «О Господи, я учу джуна делать free-to-play...»

ПРОДЮСЕР: «Интересно, где бы я сейчас был, если бы не работал над free-to-play-играми?»

Профессия: гейм-дизайнер

Чем занимается гейм-дизайнер

Гейм-дизайн начали серьезно изучать не так давно. К сожалению, научных работ, посвященных видеоиграм, тем более на русском языке, катастрофически мало. Новые материалы выходят стремительно и так же стремительно устаревают, а большие труды за редким исключением либо носят очень обобщенный характер, либо слишком сложны для новичков.

Кто такой гейм-дизайнер? Когда мы говорим о концепт-художнике, мы имеем в виду человека, рисующего скетчи; программист C# пишет код на данном языке в определенной среде разработки, сценарист работает над интересной историей – круг обязанностей большинства специалистов более или менее понятен. А чем, собственно, занимается гейм-дизайнер, когда «придумывает игру»?

Самое часто встречающееся заблуждение – гейм-дизайнер «придумывает миры», описывает лор^[12] игры, ее сюжет и персонажей. Напоминает просьбу «рассказать сказку». Придумать историю на сон грядущий, наверное, худо-бедно способен каждый. Спроектировать свою игру так, чтобы в нее было интересно играть другим, – задача куда более комплексная.

Во-первых, сюжет и сеттинг^[13] – вовсе не единственные составляющие успеха игры. Действительно, для сюжетной игры важны погружение, атмосфера, развитие персонажей, но игрой она станет, только когда в нее можно будет поиграть. Здесь работают совсем другие механизмы, нежели в книгах или фильмах.

Во-вторых, считается, что гейм-дизайнер придумывает набор правил и механик, их взаимодействие. На самом деле этим обычно занимается уже ведущий гейм-дизайнер, человек с большим опытом, или же продюсер проекта. Обычно рядовой гейм-дизайнер работает над какой-то конкретной механикой или единицей контента, а

комплексными системами занимаются старшие и ведущие специалисты. Прежде чем создать реально работающие, увлекательные механики, придется много играть, думать, анализировать и пробовать. Да и слово «придумывает» не всегда корректно. В основном это работа, основанная на анализе конкурентов, логике и математике.

Гейм-дизайнер работает непосредственно с геймплеем (игровым процессом), стараясь сделать его интересным. Этот человек всю свою карьеру думает о том, из чего складывается это «интересно» и какие конкретные инструменты применяются в самых разных жанрах и игровых ситуациях. Наличие ряда готовых решений для того, чтобы добиться этого таинственного ощущения, отличает опытного гейм-дизайнера от новичка.

Профессия гейм-дизайнера представляется людям со стороны чем-то творческим, легким и веселым. Все так, но это очень непростая работа, для которой требуются знания в самых разных областях, а также очень много упорства и трудолюбия.



Рис. 4. Кто такой гейм-дизайнер?

Навыки гейм-дизайнера

Давайте разберем, какие навыки будут полезными для становления молодого гейм-дизайнера.

ХАРД-СКИЛЛЫ^[14]

Начнем с конкретных полезных умений специалиста: знание языков программирования (C++, C#...), навыки работы с игровыми редакторами (Unreal, Unity, другие), владение иностранными языками (английский, китайский...), навыки рисования (2D, 3D, анатомический рисунок...), опыт работы с Photoshop или любым другим редактором изображений, знакомство с технологиями продаж, знание Excel, Confluence, Jira и другие умения.

Гейм-дизайнеру будет довольно сложно без знания математики. Базовая арифметика, теория вероятности и другие расчеты нужны практически на любом игровом проекте, особенно для системных и технических дизайнеров и людей, занимающихся балансом.

Эти знания необходимо также структурировать и формализовать. Полезный инструмент для этого – Excel. Придется иметь с ним дело с первого дня работы, поэтому, если почему-то вы до сих пор не пользовались этой программой, нужно наверстывать упущенное.

Важно научиться представлять, о чем думают ваши игроки – люди разного возраста и пола, со своими культурными особенностями, часто с противоположными предпочтениями. Что заставляет возвращаться в игру, какие формы и цвета стимулируют к покупке – и еще миллион вопросов, на которые предстоит ответить гейм-дизайнеру, работающему над игрой. Чтобы понять, о чем идет речь, полезно ознакомиться с дневниками разработчиков, на живых

примерах объясняющих, почему то или иное гейм-дизайнерское решение было принято, изменено или признано неудачным.

СОФТ-СКИЛЛЫ^[15]

Это ваши коммуникативные, лидерские, командные и прочие навыки: можете ли вы грамотно доносить и отстаивать свои идеи, приятно ли с вами общаться, приходите ли на работу вовремя, умеете ли сохранять холодный ум в стрессовой ситуации. То есть не то, что вы делаете, а то, как вы подходите к своим задачам.

Гейм-дизайнеру (как основному источнику задач на проекте) важно не только корректно сформулировать задачу в своей голове, но и донести ее суть до исполнителей (программистов, аниматоров, тестировщиков, художников и других специалистов). Желательно сделать это так, чтобы сохранить приятную рабочую атмосферу.

Между задачей и ее реализацией стоит много барьеров. Это понимание автором задачи ее сути, осознание того, что он хочет получить в итоге. Далее, это правильная и подробная постановка задачи и ее верное прочтение исполнителем. Но и этого недостаточно: важно еще убедиться, что и исполнитель, и автор понимают ее суть одинаково.

Знания общей психологии и социологии тоже полезны для гейм-дизайнера, так как позволяют быстрее разобраться в потребностях своей аудитории, построить отношения в команде, набрать эффективных сотрудников. Люди, в обычной жизни привыкшие анализировать свои и чужие переживания, имеют определенное преимущество.

Игровая индустрия во многом базируется на коммуникациях, и то, как вы ведете себя в команде, – критично. Проходя собеседование в игровую студию, вы с 99 %-й вероятностью столкнетесь с тем или иным заданием, цель которого – понять, как вы будете вести себя в коллективе. Все компании, проекты и задачи – разные, и

единственно верных ответов нет, но этому всегда уделяют пристальное внимание.

Игровой проект – это комплексная работа, и не всегда получается любить каждую ее часть. Хороший гейм-дизайнер работает как раз над постоянным улучшением игры, и он не должен позволять себе неконструктивно ругать то, над чем трудится вся команда. Нельзя просто заявить, что проект ужасен и ничто его не спасет. От гейм-дизайнера ждут план по постоянному совершенствованию проекта, а не пустой критики.

ИГРОВОЙ ОПЫТ

Отдельным пунктом нужно выделить игровой опыт и понимание особенностей игровой индустрии, так как это самое важное для гейм-дизайнера. Естественно, гейм-дизайнер должен любить игры и играть в них. Найдется немало людей, посвящающих много времени видеоиграм, но каждый раз подмечать закономерности, математические модели, предугадывать влияние того или иного изменения в игре могут не все. Чем больше вы знаете об индустрии, следите за тенденциями, читаете специализированную литературу, тем шире игровой кругозор и тем быстрее и лучше вы сможете оценивать свою работу и работу будущих коллег. Важно иметь игровой опыт в самых разных жанрах, следить за новинками, знать успешные и провалившиеся игровые компании, иметь представление о различных игровых мероприятиях.

Одна из ошибок гейм-дизайнера – играть только в те игры, которые нравятся. Очень часто можно найти гениальные идеи для экшена, скажем, в гонках. На другой платформе, в другом жанре вы можете почерпнуть неожиданные решения, которые еще никто не применял. Например, многие уже классические элементы шутеров заимствованы из RPG^[16], такие как характеристики оружия или

древо развития. В частности, много всего из RPG взял подкласс лутер-шутеров (*Destiny, The Division, Borderlands*).

Попробуйте для начала просто вдумчиво поиграть в любую игру, запишите свои реакции и эмоции от каких-то моментов: что напрягало, что вызывало положительные эмоции; проанализируйте свои впечатления и сравните их с впечатлениями других игроков. Чувство удовольствия от выстрела в голову соперника (хедшот) в последнем раунде, негодование от проигрыша, когда компьютеру выпала редкая карта, запотевшие ладони в момент ожидания соперника...

Может быть, вам хотелось бы смешать механики нескольких игр? Или, напротив, вы видите «лишние» игровые активности, портящие, на ваш взгляд, игру? Важно для начала научиться разбираться хотя бы в собственных эмоциях и ощущениях от игрового процесса. Такие наблюдения пригодятся при создании собственной игры, ведь многие моменты будут повторяться, помогая определить архетипы поведения игроков и полученные ими конкретные эмоции и опыт. Плюс с практикой вы научитесь и фиксировать свои субъективные ощущения, и ставить себя на место других, что приблизит вас к пониманию разных аудиторий.

Карьера гейм-дизайнера

Обязанности гейм-дизайнера в разных компаниях могут отличаться. Нет, они обязательно будут отличаться! То, что мы делаем, определяется проектом, принятыми в компании подходами и видением конкретных людей. Чем меньше команда и проект, тем больше разных ролей совмещает в себе один гейм-дизайнер. Но нужно помнить, что существует множество отдельных направлений: левел-дизайн, нарративный дизайн, технический дизайн и другие.

Где-то вы будете только конфигурировать продукт, где-то писать тонны документации, в третьей компании – создавать геймплей, в четвертой – все вместе. Но так или иначе, любой гейм-дизайнер должен уметь генерировать задачи для других специалистов с целью улучшить игру. И, как вы уже наверняка поняли, карьера будет зависеть не только от ваших усилий и личных качеств, но еще и от конкретной компании.

Что собой представляет карьера гейм-дизайнера? Большинство тех, кто идет в игровую индустрию, – люди творческие, для них самореализация и личное развитие – не пустые слова. Поэтому, говоря о развитии, важно понимать, что оно бывает не только карьерным, но и профессиональным. Их еще называют вертикальным и горизонтальным ростом.

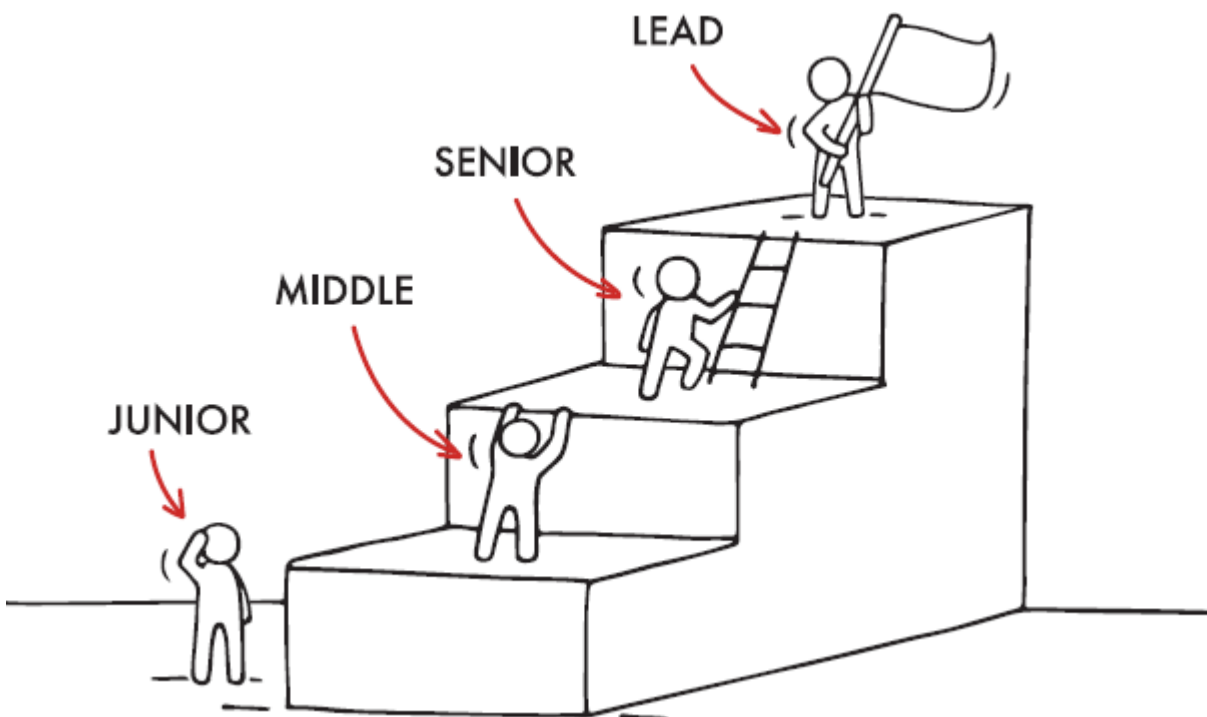


Рис. 5. Карьерный рост гейм-дизайнера

Под карьерным развитием мы будем понимать то, что называют продвижением по службе: рост должности, переход к управлению коллективом, расширение возможностей и ответственности за принятие решений. Профессиональному развитию оставим прокачку навыков в своей области, рост скорости, качества, объема знаний, а главное – глубины понимания своего дела. Такое деление отчасти условно, ведь рост по карьере зачастую невозможен без профессионального роста. А окрепшие навыки дизайна и расчетов подталкивают к повышению.

ГЕЙМ-ДИЗАЙНЕР-НОВИЧОК (JUNIOR^[17]) обычно отвечает за небольшую область, выполняя задачи, поставленные старшими товарищами, но все равно может ставить ТЗ^[18] художникам, тестировщикам, программистам и другим сотрудникам (о том, как составляются технические задания, мы расскажем дальше). Junior заводит игровые сущности, механики, логику, пишет описания, но редко придумывает что-то новое. Это работа над гейм-дизайном

именно контента^[19], хотя и здесь вы можете иметь определенную свободу в принятии локальных решений. Набирая новичков, студии предполагают их рост как специалистов, так что, если у молодого сотрудника возникла идея по развитию игры, хорошей практикой считается дать ему высказаться. Ключевое отличие работы начинающего сотрудника: к нему чаще всего приставлен ментор, проверяющий любую проделанную работу.

Для **ГЕЙМ-ДИЗАЙНЕРА СРЕДНЕГО УРОВНЯ (MIDDLE^[20])** большой объем работы – это документация, особенно если мы говорим о крупной компании. Такой специалист часто присутствует на собраниях, участвует в важных дискуссиях и имеет возможность донести свою идею о том, в какую сторону должен развиваться проект.

Получив необходимый опыт, гейм-дизайнер в состоянии принимать уже какие-то системные решения. Такой специалист может отвечать, например, за дизайн целого класса: это и внутриигровые задания (квесты), и абилки^[21] персонажей, баланс с другими классами, характеристики и прочее. Или он может быть ответственным за какую-то значимую систему – допустим, систему характеристик. Он уже может ставить задачи младшим гейм-дизайнерам (например, завести определенную абилку для класса «Паладин»), имеет представление о приоритетах и дедлайнах. Грамотно сформулированной задачи такому специалисту уже достаточно, проверять его работу нет необходимости.

СЛЕДУЮЩИЙ УРОВЕНЬ (SENIOR^[22]) – гейм-дизайнер игры в целом или же большой системы в рамках проекта. Такой сотрудник берется за выполнение самых сложных задач: ему достаточно базовых вводных данных, он сам в состоянии выяснить все подробности и проследить, чтобы работа стартовала и была выполнена в срок. Например, в его ведении может оказаться вся монетизация проекта или же он может быть дизайнером всех систем прокачки. Такие люди часто распределяют задачи по внедрению

новых особенностей проекта по отделам, и нередко их работа определяет успех игры в целом. Рутинной работы на этом этапе становится меньше; гейм-дизайнеры самостоятельно придумывают фишки^[23], доносят свои идеи до руководства и ведущих гейм-дизайнеров, совместно с проектным менеджером решают, как и когда лучше воплощать их в жизнь. Здесь, помимо опыта, на первый план выходят коммуникативные навыки сотрудника – умение договариваться и отстаивать свою точку зрения.

Есть еще **ВЕДУЩИЙ ГЕЙМ-ДИЗАЙНЕР (LEAD^[24])**. Если у геймдизайнера действительно большой опыт, он может определять направление развития игровой студии, выбирая или придумывая идеи новых игровых проектов. В крупных компаниях можно дорасти до директора направления, в том числе связанного с гейм-дизайном (например, стать креативным директором или директором по продуктам). Такой человек – управленец и нередко визионер, ответственный за реализацию игры в рамках заданной идеи и желаемых ощущений, плюс он выстраивает рабочие процессы. Удался или нет проект в целом – зона ответственности ведущего гейм-дизайнера.

Ясно, что деление это довольно условное и в разных компаниях и на разных проектах оно может быть реализовано по-своему. Но если попробовать обобщить, то путь гейм-дизайнера – это путь от исполнителя к идейному вдохновителю.

Во многих игровых компаниях принята своя должностная сетка и своя модель управления. Бывают студии с линейным управлением, когда все подчиняется узкому кругу лиц. Есть компании с проектным управлением, когда есть проектный менеджер и продюсер, в подчинении которых работают лиды и их сотрудники.

Для игрового издательства удобно иметь функциональную структуру, в которой есть отделы закупки трафика, создания креативов, комьюнити-менеджмента и т. д. и их начальники. В относительно больших студиях удобно использовать матричную структуру, сочетающую в себе проектных и функциональных

руководителей. Мы не будем рассматривать конкретные должности, но посмотрим, в какую сторону нужно развиваться для карьерного и профессионального роста.



Рис. 6. От исполнителя к вдохновителю

В первую очередь это ответственность. Чем больше зона и глубина вашей ответственности, тем выше, скорее всего, ваша должность. Новички отвечают обычно только за собственные задачи. Мидлы и сеньоры имеют в зоне своей ответственности уже целые фичи или направления. У них в подчинении может быть несколько начинающих гейм-дизайнеров. Ведущий гейм-дизайнер, как

правило, руководит всей командой гейм-дизайнеров на проекте. Он является основным источником всех задач и локально контролирует их исполнение. Если продюсер задает общее видение, то ведущий гейм-дизайнер отвечает за его реализацию. Он должен давать людям конкретику и уметь оценивать результат их работы.

Помимо расширения ответственности, мы также не должны забывать о росте профессиональных навыков. Чем больше зона нашего влияния, тем больше нам нужно знать, чтобы эффективно преодолевать стоящие перед нами вызовы.

	Задача	Фича	Проект	Компания
Стратегические		Expert	Producer	Head of Studio/ Department
Межпроектные	Senior			Head of Division
Проектные	Junior/ Specialist	Specialist/ Senior	Project Manager/ Lead	Expert

Таб. 1. Зоны ответственности по должностям

Ну и куда же без навыков коммуникации? Новые должности всегда требуют от нас лучше взаимодействовать с людьми, внимательнее слушать и быстрее принимать решения. Замкнутость и постановка задач только через таск-трекер^[25] – не лучшее решение. Например, арт-директор попросил художника нарисовать дракона для фэнтезийной игры в жанре action-RPG. Задача поставлена, дополнительных вопросов не возникло, и можно спокойно ожидать результатов. Как же руководитель удивился, когда через неделю художник показал ему механического дракона-паука! «Что ты нарисовал?!» – «Дракона, – ответил художник. – Ты же не указал деталей, а я вижу дракона именно так». Это история о сломанной

коммуникации. О том, как важно детально описывать все нюансы работы в техническом задании. И о том, что мы должны уметь говорить друг с другом.



Рис. 7. Проблема в коммуникации

Работа гейм-дизайнера в инди-команде и игровой компании

Очевидно, что подходы к постановке и реализации задач для крупных компаний и инди-студий могут сильно отличаться. Как и организация рабочего пространства, процесса разработки, взаимодействия между сотрудниками и всего, что сопутствует нашей работе. И это нормально. Эффективность требует от нас адаптироваться под текущие реалии: под людей, которые с нами работают, под собственные возможности и бюджеты и, конечно, под рынок.

Но иногда получается, что крупная компания живет по законам, существующим во многих инди-командах, и это может привести к проблемам. Большинство студий не рождаются гигантами. Они вырастают из тех самых инди. Стремительно развиваясь в финансовом плане, игровые компании, как и многие другие сферы бизнеса, часто не успевают перестроить свою административную структуру под новые реалии. Стремясь к развитию, студии расширяют штат, наращивают продуктовый опыт и полагают, что ключ к успеху – это новые и новые продукты. Больше игр – больше денег. Круче игры – больше денег. Это действительно так. Но не совсем.

В современной высококонкурентной среде построить эффективную коммуникацию для сотен сотрудников без грамотного менеджмента невозможно. Процессы, которые работали для сплоченной команды из пяти энтузиастов, неприменимы к большому проекту, и нужно вовремя суметь перестроиться.

ИНДИ-ОДИНОЧКА

Стереотипный инди-разработчик просыпается в обед, подкрепляется «Дошираком», заливается тонной кофе, после чего настает время поработать ради своего успеха. Продуктивный начальный этап генерации идей, полный мотивации и надежды, – позади; осталось самое сложное – поэтапно реализовывать задуманное. Даже для самого организованного человека на этом этапе важен план разработки.

Допустим, сегодня вы решили проработать систему фракций в вашей игре. Вы можете описать, как она должна работать, после чего, например, обратиться к похожим играм, чтобы проанализировать, как фича реализована на других проектах. В одиночку вы начинаете программировать, если не хватает артов – берете готовые ассеты^[26], рисуете интерфейс и, убедившись, что нововведения как-то работают и не сломали игру, ложитесь спать. Тестировать все, по сути, будут уже ваши игроки, а исправлять недочеты и баги^[27] в процессе вам предстоит, опять же, в одиночку.

СОТРУДНИК ИГРОВОЙ КОМПАНИИ

Просыпаться приходится раньше, чтобы погулять с собакой, отвезти детей в садик, успеть привести себя в рабочее состояние и доехать, например, через всю Москву до станции метро «Аэропорт», мечтая о корпоративном завтраке с вкусным кофе. Здесь ждут и другие прелести офиса: мягкое кресло, два монитора, спортзал, на который все равно частенько не хватает времени и сил. Прежде всего гейм-дизайнер обращается к почте, корпоративным мессенджерам (чаще всего это Skype, Slack или Telegram), Jira (или другой программе, где ставятся задачи) и своему календарю (чтобы вспомнить о запланированных собраниях, плейтестах^[28] и прочих организационных моментах).

Прежде чем принять поставленную задачу, естественно, необходимо с ней ознакомиться, убедиться, что никаких вопросов

она не вызывает, в противном случае связаться с заказчиком. После чего задачу предстоит оценить, то есть обозначить сроки выполнения.

Проектный менеджер (ПМ), ссылаясь на план, уточняет, готовы ли вы сегодня наконец приступить к созданию фракционной системы. Так как еще 60 человек ждут проработки этой фичи, выбор невелик: команду подводить нельзя. Так что вы приступаете к анализу конкурентов, составлению документации, сбору обратной связи от коллег, чтобы в середине дня отправить результаты своей работы продюсеру проекта. Последний вносит свои правки, и вам предстоит их проработать, после чего документ принимают, а ПМ распределяет задачи по отделам, откуда немедленно поступают вопросы, что же вы имели в виду. Дав необходимые консультации, вы понимаете, что время – 17:00, вы очень устали и должны срочно выпить кофе. Иногда по результатам очередного общения с ПМ вам приходится задержаться, чтобы закончить свою работу. После чего с чувством выполненного долга можно отправиться домой, а на следующий день проделать все то же самое, так как продюсер посчитал, что конъюнктура рынка изменилась и фракционная система больше не нужна, а вот клановые бои необходимы немедленно.

Если вы работаете в большой игровой студии, скорее всего, вам придется смириться с тем, что много сил и времени вы будете тратить на работу «в стол». Программисты пишут код для забракованных впоследствии прототипов, художники создают сотни артов, из которых выбирают только один, гейм-дизайнер бесконечно переписывает документацию, исходя из нового понимания рынка продюсером.

Конечно, мы немного драматизируем. Можно подумать, что продюсер – какой-то злодей или садист, но правда в том, что именно этот человек несет ответственность за качество продукта. Вот почему ни в коем случае не надо совмещать в одном лице функции продюсера и проектного менеджера. Задача первого – сделать игру максимально качественно, второго – не выходить за

рамки сроков и ресурсов. В некоторых игровых студиях продюсер считается главным на проекте, так как он отвечает за деньги, прибыль проекта; далее следует ПМ, которому подчиняются лиды направлений.

Таким образом, с одной стороны, продюсер не влияет на работу каждого специалиста самостоятельно, он не должен ставить задачи сотрудникам в обход проектного менеджера. С другой стороны, как ответственное лицо и руководитель проекта именно продюсер принимает ключевые решения. Получается, что продюсер делится своим опытом и видением проекта с командой, но не вмешивается в процесс. Если продюсер видит, что необходимо внести изменения, сначала он должен обсудить с ПМом, насколько это увеличит срок разработки. ПМ, в свою очередь, не может решать продуктовые вопросы – это зона ответственности продюсера.

Особенности работы в игровой студии

Ради чего люди идут работать? Самый простой и понятный вариант – чтобы получать деньги, и таких людей легко мотивировать премиями.

Игровые компании знают, что их сотрудники очень ценят, например, комфорт. Поэтому они стараются, чтобы люди работали в уютном офисе с диванчиками и пуфиками, предоставляют завтрак (кофе, фрукты, печенье), стараются сделать атмосферу уютной за счет пледов, настольных игр, приставок, в которые можно поиграть с коллегами после тяжелого трудового дня, и пр.

Другая важная вещь – это возможность реализовать свои идеи в рамках создаваемых игр. Работая в игровой индустрии, вы можете проявить инициативу, продемонстрировать нешаблонное мышление и креатив.

Большинство студий стараются уделять много времени корпоративным связям. Обычно любой сотрудник может предложить свои идеи старшим товарищам.

Многие идут работать в игровую компанию, чтобы получить опыт, необходимый для реализации собственных проектов (в качестве инди-разработчика или открыв свою студию).

Как показывает практика, попасть в игровую индустрию мечтают многие сотрудники крупнейших компаний, работающих в других отраслях. Причем совершенно разных и порой даже не связанных с IT: это и банковский сектор, и юриспруденция, и журналистика, и многие другие сферы деятельности – как технические, так и гуманитарные.

Главное отличие работы в игровой компании в том, что человек занимается тем делом, которое его увлекает и ради которого он каждый день с энтузиазмом ходит на работу. По собственному опыту можем сказать, что когда работаешь с удовольствием, то финансовая стабильность сама находит тебя.

Игровой рынок активно растет и останавливаться в своем росте в ближайшие годы не планирует. Это позволяет сотрудникам игровых компаний не только работать над интересными и творческими задачами, но и чувствовать себя финансово обеспеченными.

Препродакшен

Этапы разработки игрового продукта

Игровая индустрия позаимствовала у кино названия для основных этапов разработки: **ПРЕПРОДАКШЕН, ПРОДАКШЕН И РЕЛИЗ**. Для многих игровых проектов не менее важными становятся этапы **ПРОТОТИПИРОВАНИЯ** (цель которого – понять, какой проект в итоге делать) и **ОПЕРИРОВАНИЯ** – это время поддержки и развития успеха готового продукта на рынке.

ЭТАП	ИТОГ
Концептирование (Concept)	Концепт-документ/Вижн
Прототипирование (Prototyping)	Прототип
Вертикальный срез (MVP)	Альфа
Производство контента и дополнительных фичей (Concept Production)	Закрытое тестирование (Close Beta Test / Friends and Family Test)
Последние фичи, багфикс, фиче фриз	Открытое тестирование (Open Beta Test / Soft Launch / Early Access)
Доработка по итогам открытого тестирования	Релиз

Таб. 2. Этапы производства игры

Длительность каждого из этапов зависит от нужд конкретного проекта. Использование инновационной механики потребует от вас большого количества прототипов; игры, сильно зависящие от маркетинга и работы с комьюнити, потребуют много сил на этапе релиза; оперирование проектом очень важно для многопользовательских проектов, предполагающих регулярный выпуск обновлений и микротранзакции.

Не всегда компания-разработчик и компания-оператор (компания, занимающаяся оперированием проекта) объединены в рамках одной команды. Во-первых, вы можете работать над игрой для внешнего издателя, с которым заключается договор, где указано, что, в какие сроки, в каком качестве и за какое вознаграждение должно быть сделано. У такой команды нет комьюнити-менеджеров, сотрудники не будут заниматься маркетингом игры – все это ложится на плечи издателя.

Другой вариант: продюсер выступает в роли издателя, самостоятельно контролируя разработку, но оперирование проекта поручается внешней компании. В некоторых случаях решающее слово может быть не столько за продюсером, сколько за владельцем студии, за человеком, владеющим бюджетом компании.

Третий вариант – одна команда и разрабатывает, и продвигает игру. В этом случае продюсер принимает все ключевые решения, руководит не только разработкой, но и оперированием (трафик, маркетинг, распределение финансов и т. д.).

Как мы помним, игровая индустрия – это бизнес, связанный с большими рисками. С точки зрения разработки это риск не сделать игру в прогнозируемые сроки, риск не успеть довести игру до необходимого уровня качества, риск, что продукт не будет востребован аудиторией, что мы не сможем замотивировать и удержать команду, и прочее. Поэтапный подход позволяет минимизировать подобные риски.

Построение процессов на этапе препродакшен

КЛЮЧЕВЫЕ ЦЕЛИ ЭТАПА ПРЕПРОДАКШЕН

Итак, что мы должны сделать на этом этапе:

- научить команду слаженно работать;
- выстроить процессы разработки;
- создать первичную документацию (вижн, концепт, feature-list (список фичей), art-style (арт-стиль), бюджет, бизнес-план, проектный план, roadmap^[29]);
- проверить все ключевые идеи прототипами.

Пока идея обрастает подробностями, гипотезы проверяются на прототипах, четко не определен графический стиль и так далее – вы находитесь на этапе препродакшен. Начинающему разработчику сложно предсказать, сколько времени он займет, ведь чтобы это сделать, нужно иметь опыт во всех областях. Потребуется собирать информацию о сроках у ваших специалистов, либо же рассчитать сроки вам поможет ПМ, предварительно пообщавшись с другими сотрудниками.

КАК ВЫБРАТЬ ИДЕЮ ИГРЫ

Часто начинающие разработчики боятся делиться своими идеями публично или даже с близкими, опасаясь, что кто-то может украсть их уникальную концепцию игры. Вынуждены огорчить: сама по себе идея не стоит ничего. Вообще. В отрыве от методов и стоимости реализации, без играбельного прототипа ваша идея никому не нужна.

Чаще всего что-то подобное уже создавалось, и гейм-дизайнеры пытаются изобрести велосипед. Или же идея действительно

великолепна, но ее реализация настолько дорогостоящая, что никто не решается воплотить ее в жизнь. Обратная ситуация, когда разработчик даже не пытается придумать что-то свежее, потому что не верит в то, что это возможно, – тоже проигрышная стратегия.

В игровой индустрии все взаимосвязаны, люди обмениваются опытом и идеями на конференциях, используют механики чужих игр, чтобы улучшить свою, обсуждают на форумах те или иные решения, дают советы, так что не бойтесь за свою идею. Лучше приступайте к ее реализации!

На самом деле для создания игры вам потребуется сотня идей. Нужно уметь генерировать их, обсуждать, принимать решение и, не жалея, отказываться от неудачных вариантов, даже если вы уже довольно много сделали, чтобы их воплотить. Игровая индустрия – крайне динамичная область, и гибкость здесь необходима.

Существует множество подходов, как развить креативность и какие методы лучше использовать в работе творческой команды: «брейнштурмы», «шесть шляп», – вы сможете найти самые разные методики и выбрать ту, что подходит именно вашей команде. Главное – не бояться, что идея окажется не так хороша, а реализация ее займет долгие годы или что у вас не хватит возможностей и ресурсов, чтобы ее воплотить. Начинать всегда стоит с простого: записать правила, создать прототип на бумаге или в PowerPoint, продумать мотивацию для игроков; первый этап одновременно самый легкий (с точки зрения реализации) и самый сложный, так как перенос своей идеи в реальность – нетривиальная задача.

Сначала у вас как гейм-дизайнера – полная свобода действий и решений. Цель: последовательно накладывать ограничения, чтобы в итоге иметь четко заданные правила и параметры, делающие игру «интересной». Хорошие игры в открытом мире, которые любят за «полную свободу действий и передвижений», на самом деле череда ограничений взаимодействия с игровой реальностью, задающая цели и подталкивающая к исследованию и принятию решений.

Путь от «хочу придумать игру» до восторженных отзывов игроков лежит именно через ограничения. Сколько игроков должно играть? Реализм или фэнтези? Будет ли рандом существенно влиять на процесс? Сколько длится игровая сессия?

Множество решений гейм-дизайнеры принимают сами, но есть и требования извне: игроки хотят игру по известной франшизе, инвестор готов выделить средства только на четыре месяца разработки, новое поколение консолей диктует технические требования, растет популярность определенного жанра и прочее. Это кажется парадоксальным, но именно рамки заставляют нас двигаться вперед. Так что если вы не знаете, с чего начать, или где-то застряли, именно новое ограничение может помочь продвинуться в работе над игрой. Хрестоматийный пример – туман в *Silent Hill*: он стал символом серии, хотя вырос из технических ограничений прорисовки и желания создавать просторные городские локации.

Вдохновение необходимо черпать прежде всего из других игр. Гейм-дизайнеры стараются играть не только в свои любимые жанры; знание множества самых разных механик, изучение документации и известных и свежих игровых проектов в итоге дают базу для появления необычного сочетания элементов игры. Общение с другими гейм-дизайнерами также стимулирует более глубокое осмысление игрового процесса и вдохновляет на создание чего-то нового.

Идея может базироваться на мифологии, истории, предметах искусства, кинематографе, актуальной проблеме современного мира, другой игре – неважно. Далеко не все игроки вообще замечают общую тему игры, потому что та, например, обращена к их подсознанию или же для них достаточным определением является «интересно-неинтересно». А что он под этим подразумевает – головная боль гейм-дизайнера: исследование мира или приятные ощущения от преодоления трудностей; собственные ассоциации и фантазии или драматический сюжет; а может быть, он ценит возможность общения с другими игроками?

Так или иначе, чтобы игрокам все-таки было «интересно», гейм-дизайнер должен не только обозначить общую тему, но и использовать все возможные средства для ее реализации. Так называемая «креативность» гейм-дизайнера – это способность повсюду замечать и фиксировать опыт и эмоции, которые впоследствии он сможет преобразовать в конкретные игровые фишки, делающие игру запоминающейся и уникальной.

Попробуйте подумать над потенциально успешной игрой, в которой вам «чего-то не хватило». Как можно было бы ее улучшить?

В жизни есть множество систем, стимулирующих нас принимать решения (отношения, хобби, карьера...) – их также можно брать за основу для игры. На этом этапе не так важен выбор сеттинга или игровой платформы, вам предстоит определить ощущения, которые вы хотите передать.

Чем раньше вы определитесь с основной идеей игры, тем лучше – единая цель поможет быстрее понять, какие элементы необходимы для ее осуществления, а какие второстепенны или не нужны.

Любой концепт предполагает получение различного вида опыта, так что вам предстоит подробно прописать его составляющие, чтобы определить конкретные пути и способы достижения предполагаемых эмоций. Определившись с основной темой и сеттингом, часто гейм-дизайнеры погружаются в доскональное изучение всевозможных источников информации, на основании чего позже и формируется дизайн игры. Проработка деталей – еще одна важная задача гейм-дизайна.

Итак, идея в отрыве от реализации не представляет большой ценности. Не надо бояться делиться своим опытом и изучать чужие решения. Ограничения помогают конкретизировать желаемые ощущения от игры и общую идею.

ДОКУМЕНТАЦИЯ: КОНЦЕПТ

На начальном этапе необходимо сформировать **КОНЦЕПТ-ДОКУМЕНТ**. Именно с него начинается реальная работа над проектом. В идеале это одна страница, где суть проекта отражена таким образом, чтобы любой человек мог за несколько минут понять, что собой представляет игра. Задача – продать ваши идеи, то есть преподнести их так, чтобы команда захотела это делать, инвестор вложил в это деньги, потенциальные игроки захотели в это сыграть. Вам просто нужно объяснить, почему это будет круто, подобрав формулировки для разных целей.

ОБЫЧНО КОНЦЕПТ ВКЛЮЧАЕТ В СЕБЯ

КРАТКОЕ ОПИСАНИЕ ИГРЫ: игровая платформа, жанр, сеттинг, модель распространения, описание базового геймплея, основные игровые механики и цели игрока.

Нередко, если после устного обсуждения попросить каждого из членов команды составить такой документ, вы получите описание нескольких разных игр. Возьмите все лучшее из каждого и, собравшись еще раз, утвердите финальную версию.

РЕФЕРЕНСЫ: похожие или вдохновляющие игры, примеры графического стиля, атмосферы, музыкального сопровождения, удачно реализованных игровых механик и прочее.

Конкретные примеры нужны для того, чтобы исключить разное понимание составляющих вашего проекта. Это могут быть настольные игры, картины, фильмы, комиксы, книги, конкретные механики других игр и т. д.

Референсы предлагают набор готовых решений. Сравнив с ними ваши идеи, вы сможете быстрее понять, в каком направлении двигаться. Именно так вы проверяете собственную идею и уменьшаете время на коммуникацию с другими людьми, избегая ее повторного описания.

АНАЛИЗ КОНКУРЕНТОВ: само по себе словосочетание «анализ конкурентов» ничего не означает. Чтобы он был эффективным, необходимо составить список гипотез, которые мы хотим проверить, ведь любая аналитика должна иметь конкретные цели и заказчика.

Итак, есть игровой продукт: определенный жанр, включающий в себя набор механик, подходящий арт-стиль, технические решения, сеттинг и пр., – и есть аудитория, любящая подобные игры. Чтобы понять, кто наши конкуренты, прежде всего необходимо выделить игры с одним или несколькими аналогичными параметрами. Допустим, мы работаем над мобильной матч-3 с сюжетом и не можем определиться с сеттингом. Составив список подобных игр, мы сможем создать статистику, отражающую выбранные конкурентами решения, количество играющих пользователей, доходы игры и другие показатели.

Заказчиком анализа конкурентов могут быть разные специалисты. Программиста, к примеру, может интересовать статистика характеристик различных устройств, чтобы выделить целевые и определиться с выбором игрового движка; очевидно, что без помощи художника не получится самостоятельно выбрать арт-стиль игры.

Просто выписать всех конкурентов недостаточно. С помощью конкретных специалистов необходимо составить классификацию, зачастую отдельно для каждого проекта, после чего аналитик сможет подготовить отчет для принимающего решение человека, чаще всего продюсера.

Ситуация может быть и обратной: допустим, наша команда умеет делать игры только в определенном арт-стиле. Тогда перед нами стоит задача понять, сколько игроков заинтересуется таким проектом, сколько у нас конкурентов, падает или возрастает популярность выбранного арт-стиля, чтобы в итоге решить, имеет ли смысл создавать в нем свою игру.

Отдельно стоит уделить внимание маркетинговому анализу конкурентов: какие статьи пишут, что за слоганы и картинки

используют для продвижения, в каких выставках участвуют и т. д.

Проанализировать все стратегии или шутеры – сложновыполнимая задача, лучше сосредоточиться на 5–10 играх, с которыми вы точно будете конкурировать за аудиторию. Изучив, где размещаются конкуренты, вы сможете предположить, где искать своих потенциальных поклонников.

Но отследить, откуда пришли ваши игроки, нетривиальная задача. Есть закупка трафика^[30], позволяющая оценить эффективность тех или иных решений, а есть все остальное: работа с блогерами, статьи в игровых изданиях, ведение социальных сетей и прочее. Результат такой работы обычно отложенный, и контролировать его трудно. К примеру, может показаться, что единовременно обзор на Youtube привел мало пользователей. А позже выяснится, что этот ролик имеет стабильное количество просмотров и до сих пор приводит новых игроков, а значит, вложение себя окупало.

Анализ конкурентов помогает еще на начальном этапе разработки определиться с ключевыми особенностями и преимуществами. Однако нередко проекты, которые должны быть закончены за несколько месяцев, растягиваются на годы. И может наступить момент, когда станет необходимым еще раз проделать всю эту работу, чтобы вовремя повернуть проект в нужную сторону.

К анализу конкурентов стоит подходить исходя из имеющихся ресурсов. Если вы – крупная компания, которая может позволить себе дорогостоящую рыночную аналитику, приступайте к реализации ее итогов. Facebook Audience Insights, App Annie, Sensor Tower, AppMagic и другие средства систематизации данных способны сильно помочь, хотя и у них есть ограничения: данные могут оказаться обобщенными, а метрики, например по азиатскому рынку, все равно придется собирать отдельно.

Инди-команды чаще всего пользуются открытыми средствами аналитики, чтобы определить, есть ли смысл следовать за лидерами рынка, создавая нечто похожее, или лучше сосредоточиться на нишевой аудитории. К открытым средствам аналитики можно

отнести топы игровых магазинов, открытую аналитику от Newzoo, Superdata, Sensor Tower (базовый функционал статистики) и другие.

УНИКАЛЬНЫЕ ОСОБЕННОСТИ ИГРЫ (USP^[31]): перечислите особенности вашей игры, выделяющие ее из множества других аналогичных проектов.

Чтобы определиться с этим пунктом, необходимо представлять себе вашу целевую аудиторию и ответить на вопрос: кто он, человек, которому должна понравиться ваша игра? После чего вам предстоит определить, что конкретно в вашей игре будет «цеплять»: может быть, инновационная идея или технология, необычное сочетание привычных и любимых механик, свобода действий и т. д. «Красивая графика и интересный сюжет» – это не USP. Примером хорошего USP могут служить, например, нейросеть в «Блицкриг 3» в качестве противника, анализирующая тактику игроков и обучающаяся на этом, или продолжительность сюжетной кампании в 100+ часов в «Ведьмаке 3».

СТЕРЖНЕВЫЕ ОЩУЩЕНИЯ: определение главных, стержневых ощущений от игры, или core-геймплея^[32], делает вашу идею более реальной. Нужно предположить, за что игроки полюбят ваш проект, какие эмоции они получат, какой игровой опыт приобретут и какие потребности удовлетворят.

Потренироваться определять свои ощущения стоит на хорошо знакомых вам играх. Попробуйте декомпозировать (разделить их на составные части): какое ощущение в игре главное? в чем «изюминка»? с помощью каких механик достигается? Если не очевидно, зачем вообще играть в вашу игру, скорее всего, что-то уже идет не так.

Нет ни одной массово успешной игры, которая не работала бы со стержневым ощущением. Многие играли в *The Sims*. Что именно делает игру популярной, что дарит в ней радость? В основном это две вещи: сделать что-то экстремальное, но в приближенных к реальности условиях, или же достичь быстрого прогресса (сложные, долгие в жизни вещи, такие как карьера или отношения, мы можем

получить здесь и сейчас). Построить дом своей мечты, с красивой мебелью, собакой и бассейном, или же включить фантазию и собрать дикий лабиринт – в общем, реализовать что-то, что либо невозможно, либо слишком необычно, либо слишком долго и трудно в реальной жизни.

Таким образом концепт-документ – это первый инструмент, позволяющий зафиксировать ваши идеи и договоренности.



Рис. 8. Известное правило бизнеса

ДОКУМЕНТАЦИЯ: ВИЖН

Следующий необходимый, а может быть, самый важный документ – это **ВИЖН**. Обычно это 4–5 страниц А4, которые будут давать представление о вашем проекте, пока игра еще не готова. Вижн

описывает игру как бизнес-продукт. Берем концепт-документ и добавляем туда следующие пункты.

ЦЕЛЕВАЯ АУДИТОРИЯ

- демография;
- модель поведения (Бартл, BrianHex и другие);
- категория игроков (на какую аудиторию нацелена игра: Casual, Midcore, Hardcore-аудитория).

«ФОРМУЛА УСПЕХА»

То, что в вашей игре будет самым важным. Это необязательно что-то уникальное (как USP). Нужно просто ответить на вопрос, на чем мы делаем акцент: честная физика стрельбы, гиперреалистичная графика и т. д. Именно эту часть работы мы выделяем как приоритетную и пристально следим, чтобы она была выполнена в максимальном качестве. Часто это называют pillars («столпы») – некие незыблемые элементы, на которых держится задумка игры.

ГРАФИЧЕСКИЙ СТИЛЬ И СЕТТИНГ

Нужно определить, в каком стиле и визуальном исполнении будет игра: нуарный детектив, мультяшное фэнтези, киберпанк и т. п. Можно использовать сервисы вроде Splitmetrics. Это некая имитация стора, куда загружаются фейковые скриншоты из игры: более реалистичная графика, мультяшная, комиксовая. Так, можно посмотреть, какой вариант захочет скачать большее число потенциальных игроков, и сделать выводы о предпочтениях аудитории.

АНАЛИЗ РЫНКА

- Конкуренты и их показатели;
- размер рынка.

МОДЕЛЬ МОНЕТИЗАЦИИ

Монетизация в играх – это конвертация игроков в деньги, независимо от того, зарабатываем мы на продаже самой игры или внутриигрового контента. Выбор монетизационной стратегии зависит от особенностей проекта, региона и предпочтений целевой аудитории. Моделей монетизации довольно много, мы перечислим основные.

- **BUY-TO-PLAY** – «купи, чтобы играть». Игра выставляется в магазинах, и, чтобы получить к ней доступ, игрок должен оплатить покупку. Например, *Cyberpunk 2077* или *Ведьмак*. Такие игры часто зарабатывают также на **ПРОДАЖЕ DLC**^[33]. Это могут быть новые части любимой игры, дополнительные уровни, специальные издания и т. д. Продажа DLC более популярна для ПК и консольных игр, чем для мобильных проектов.

- **PAY-TO-PLAY (ПОДПИСОЧНАЯ МОДЕЛЬ)** предполагает, что для постоянного доступа к игровому контенту нужно оформить платную подписку. Такая модель используется, в частности, в глобальных ММО и других играх, предполагающих высокую степень вовлечения. Яркий пример – *World of Warcraft*. В последние годы подписочная модель все больше используется и вне игровой индустрии: создателями музыкального и видеоконтента, книг, других сервисов. В геймдев она возвращается в виде набирающих популярность облачных сервисов.

- **FREE-TO-PLAY** – монетизационная модель, позволяющая играть без обязательного внесения денежных средств. Такие игры зарабатывают, предлагая игрокам за деньги получить игровое преимущество, уникальные предметы, открыть новый игровой контент и т. д. Как пример можно привести *Pokemon Go*, *League of Legends*, *LineAge 2*, *Perfect World*, *Genshin Impact* и пр.

Встроенные покупки помогают игрокам продвинуться в игре: за дополнительную плату можно ускорить процесс прокачки, получить мощные игровые предметы и другие бонусы. Яркий пример – *Fortnite*. Помимо полезных для игрового прогресса вещей, часто разработчики предлагают купить чисто декоративные элементы, позволяющие внешне выделиться среди других игроков: красивые скины, прически для персонажей, уникальные костюмы и пр. Внутриигровые покупки реализуются не только напрямую в игровом магазине, но и за счет различных монетизационных акций, лотерейных механик, лутбоксов и т. д.

Несмотря на то что многие мобильные игры, особенно батлеры и стратегии, до сих пор активно продают боевое преимущество (мощь), тренд долгое время уходил от продажи силы в сторону продажи времени, позволяя игроку играть меньше, компенсируя свое отставание от лидеров за деньги. Современные игры, работающие по модели Free-to-play (F2P), уже задали новый тренд. Это продажа бонусов, не влияющих на игровой процесс. Речь идет в первую очередь о внешнем облике: скинах, костюмах, ездовых животных, небоевых питомцах, рамочках для аватаров, голосах озвучки и т. д. Эту тенденцию задали киберспортивные ПК-проекты. Их масштаб и проникновение в массовую аудиторию не могли не оставить след в потребительских предпочтениях. И, несмотря на то что мобильные игры остаются оплотом классической монетизации на микротранзакциях, все чаще появляются игры, для которых продажа силы и времени уходит в прошлое.

- **ВСТРОЕННАЯ РЕКЛАМА** – еще одна важная модель монетизации. Она позволяет разработчикам зарабатывать за счет

того, что игроку показывается реклама других игр, приложений, услуг и т. д. Реклама должна привлекать внимание, но при этом не мешать получать удовольствие от самой игры. Популярный сегмент гиперказуальных игр зарабатывает и растет в первую очередь за счет доходов от встроенной рекламы (например, *Helix Jump*). Встроенная реклама может быть интегрирована в игру, например в качестве изображения или видео, которые игрок видит при загрузке. Основная проблема для разработчиков мидкорных и казуальных игр – решить, как встроить рекламу таким образом, чтобы не разочаровать и не раздражать игроков.

В гиперказуале такой проблемы нет. Даже сами платформы, поставляющие услуги рекламодателей, уже предлагают готовые решения. Это *interstitial* – межстраничный баннер, показывающийся между уровнями или при смене экранов интерфейса. Или *underground banner* – маленькая полоска баннера внизу экрана. И конечно, *rewarded* – запрашиваемый пользователем по его собственному желанию просмотр рекламы, за который он обязательно получает обещанную награду, если досмотрит видео до конца. 97 % выручки гиперказуальных игр приходится именно на рекламу. Конечно, можно сказать, что реклама, как и *pay-to-win* (игры, где вложение реальных денег дает значительное преимущество перед остальными игроками), киберспорт и фримииум – это части F2P. И это будет верно. Но сейчас под F2P чаще понимают только микротранзакции, а перечисленные методы выносят в отдельные большие модели.

- **FREEMIUM (ФРИМИУМ)** – монетизационная модель, предлагающая игроку бесплатную ограниченную версию игры с возможностью приобрести расширенную или *premium*-версию. Отличие от классического *Free-to-play* в том, что игроку до внесения оплаты предоставляют доступ к изначально урезанному геймплею или ограниченному набору контента. Например, так происходит в *Star Wars: Old Republic*.

- **PRODUCT PLACEMENT**, то есть наличие неявной (скрытой) рекламы – тоже возможность получить прибыль. Усталость игроков от навязчивой прямой рекламы подтолкнула разработчиков игр к партнерству с различными брендами. Например, в *Death Stranding* герой может выпить энергетик Monster Energy, чтобы повысить некоторые показатели, а персонажи *Fortnite* носят кроссовки Nike.

- **ЛИЦЕНЗИРОВАНИЕ** дает разрешение на использование какого-либо объекта интеллектуального права. Его трудно назвать моделью монетизации продукта, но к монетизации аудитории оно имеет непосредственное отношение. Это может быть единовременный платеж за использование или роялти – процент от продажи продукта, использующего лицензию (иногда эти условия комбинируют). Предметом лицензии может быть что угодно – лор, персонажи, названия и т. д., но чаще всего лицензия дается на всю игровую вселенную. Можно не только разрабатывать игру специально для партнеров, которые будут использовать ее для своих целей за оговоренный гонорар, но и зарабатывать на продаже франшизы для создания новых медиапродуктов (кино, комиксов и пр.) или сувенирной продукции и аксессуаров (футболки, фигурки, постеры, товары для дома). Яркие примеры известных игровых франшиз – *Super Mario Brothers*, *Resident Evil* и др.

РИСКИ

Далее в вижн следует записать, с какими проблемами может столкнуться проект.

- Список основных рисков (от самых опасных до незначительных);
- стратегия уменьшения рисков;
- SWOT-анализ^[34].

Если идея не позволяет собрать рабочий прототип за несколько часов/дней, а, напротив, требует многомесячной работы программистов, художников и других специалистов, вы столкнетесь

с рисками, сопровождающими любую сложную разработку. Невозможно точно спланировать, сколько времени, ресурсов и тестов потребуется, чтобы достичь желаемого качества. Неверные предположения приведут к дополнительным расходам и зря потраченному времени, что легко может сделать игру убыточной, а усилия – напрасными.

Каждый прототип, брейншторм или командный анализ рисков будет приближать к более точному пониманию, что за игру вы делаете и сколько времени потребуется на ее создание.

Надеяться на удачное стечение обстоятельств – не лучшая стратегия. Просто поэтапно собирая воедино все компоненты игры, вы рискуете потратить слишком много времени на получение материалов для первого плейтеста. Лучше сразу всей командой определить, с какими конкретными проблемами может столкнуться ваша игра (интерес к механикам, технические возможности выбранного движка, время на создание анимаций и другого контента, внешние факторы), продумать, как избежать возможных проблем, и зафиксировать все это в документах.

Концепт фиксирует вашу идею, а вижн поясняет, как и почему эта идея будет работать – что должно понравиться игрокам и приносить прибыль, – а также создает единый образ итогового проекта в головах всех участников разработки. Вероятнее всего, оба эти документа будут еще меняться и актуализироваться в процессе разработки. С их помощью вы можете оценить шансы на успех и дорабатывать слабые стороны, пока не достигнете уверенности в результатах. Но делать это лучше до этапа продакшен, когда любые изменения повлекут за собой проблемы.

Концепт и вижн помогают сформировать гейм-дизайн-документ [\[35\]](#) (ГДД), о котором речь пойдет ниже. ГДД создаются и на этапе препродакшен, и на этапе продакшен.

Выбор управленческих методологий для команды

Успех игрового проекта часто зависит именно от счастья и мотивации ваших сотрудников. В более или менее крупных компаниях это сфера деятельности проектного менеджера. Гейм-дизайнеру просто нужно знать, какие модели существуют.

Когда вы собираете свою первую команду, иногда лучше заранее договориться, что проект должен быть простым и может даже ничего не заработать. Часто цель совместной работы над дебютной игрой – налаживание процессов, обучение и опыт. Можно, например, создать клон любимой игры, добавив туда одну самостоятельную фичу, или переработать сеттинг. Работая над жанром с устоявшимся геймплеем, вы можете сосредоточиться на создании контента, чтобы в будущем четко понимать, какое время занимает реализация той или иной фици.

Возможно, имеет смысл выбрать проект, главной особенностью которого является что-то, в чем вы точно сильны. Например, для создания визуальной новеллы ключевым навыком будет умение работать именно со сценарием, а не гейм-дизайн или онлайн-взаимодействие. Также всегда можно обратиться за внешней консультацией, чтобы оценить свои идеи.

Лучше уже на раннем этапе задуматься о методах управления. Здесь есть два подхода: можно сначала найти нужных специалистов и затем выбирать методологию управления конкретно под этих людей; или же вы как руководитель проекта уже знаете, как вам удобнее работать, осталось найти людей, которые разделяют ваши взгляды и готовы работать в выбранном формате. Ниже мы разберем принципы нескольких моделей управления.

WATERFALL[\[36\]](#)

Эта управленческая модель была популярна для разработки до начала 2000-х годов. Смысл в том, что очень много времени мы отводим на тщательное составление четкой документации и лишь после этого приступаем непосредственно к разработке. Проблема в том, что составить адекватный подробный план на год вперед – крайне сложная задача.

Такой подход основан на линейном, последовательном цикле разработки: концепт → составление документации → аналитика и сбор информации по рынку → дизайн → программирование → тестирование → поддержка.



Рис. 9. Цикл разработки Waterfall

Для крупных проектов, над которыми работают команды более ста человек, в приоритете качественное планирование и поэтапная работа над продуктом. Поэтому каскадные методологии до сих пор остаются актуальными – правда, этапы теперь планируются не на весь проект в целом, а в рамках повторяющихся циклов, итераций.

Главная проблема Waterfall – невозможность перейти к следующему этапу, пока не закончен предыдущий, или вернуться на предыдущий этап, не начиная процесс сначала. То есть, например, если тестирование выявило ошибку в программировании, вы должны будете сначала убедиться, не было ли этой ошибки в документации, и только потом отдавать результаты тестирования программисту. Такой метод плохо подходит для динамичной разработки, требующей молниеносных решений и внесения изменений.

AGILE

Agile – скорее концепция, нежели четкая методология. Она оформилась в 2011 году, хотя существовала и ранее. Благодаря многократному повторению этапов достигается гибкость разработки продукта. Применение Agile подразумевает тесный контакт разработчиков с бизнес-структурами. В отрыве от заказчика, без постоянных изменений, основывающихся на его отзывах или отзывах пользователей, работать будет довольно сложно.

Если следовать Agile, внесение изменений в процессе разработки намного важнее следования планам. Конечно, документации никто не отменяет, но работающий софт и прямое взаимодействие сотрудников в приоритете.

Сотрудники разбиваются на группы: либо по специализации (команда тестировщиков, команда художников, команда программистов и пр.), либо создаются страйк-команды – люди с разными навыками, объединяющиеся для создания определенной фичи. Первый вариант больше подходит для планомерной работы над игровым контентом, второй – для какой-то важной и срочной задачи, работа над которой не предполагает отвлечения на что-либо другое.

Цикл разработки в Agile выглядит так:

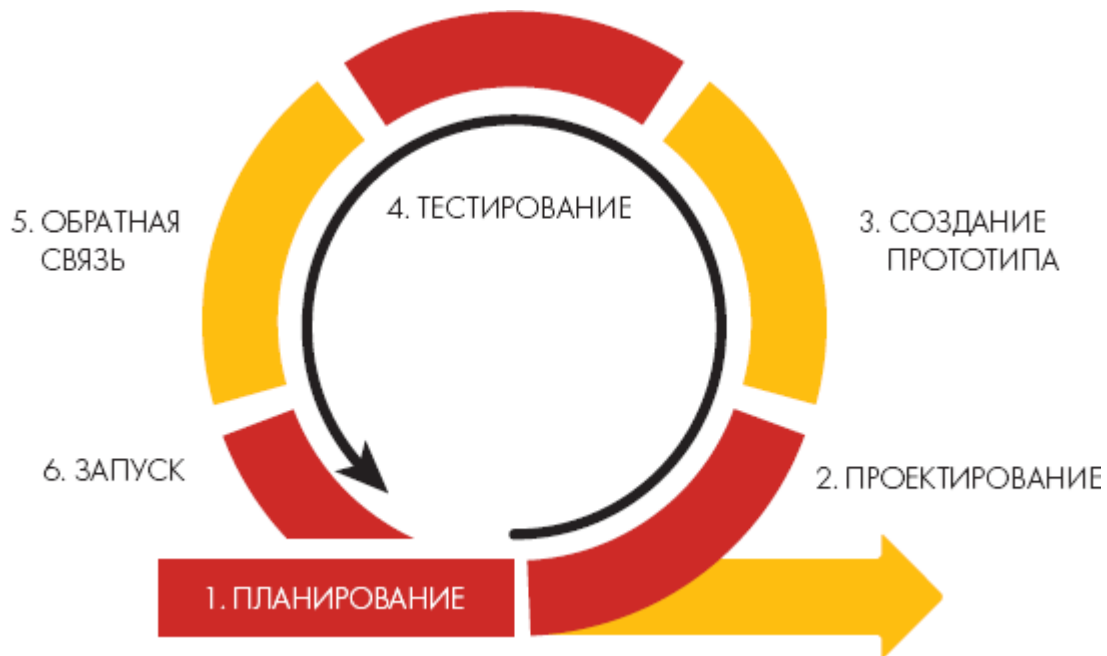


Рис. 10. Цикл разработки в Agile

И Agile, и Waterfall позволяют создавать качественные продукты. Большинство игровых компаний используют гибридные методы этих двух методологий, в чистом виде они почти не встречаются. Если у заказчика есть четкое понимание, чего он хочет, а у вас – как воплотить это в жизнь, более жесткие процессы Waterfall могут быть эффективнее. Если же игра содержит экспериментальные идеи, заказчики хотят вносить изменения, а на реализацию мало времени – обратите внимание на методы Agile.

SCRUM

Scrum – один из самых популярных способов практического внедрения философии Agile. Он подходит в основном небольшим командам до 10 человек, причем используется повсеместно, не только в IT-сфере. Предполагается, что некоторые члены команды возьмут на себя определенные роли, обязанности и будут соблюдать четыре постоянно повторяющиеся «церемонии»:

- планирование спринта^[37] (итерации);
- регулярный стенд-ап;
- подведение итогов по задачам спринта;
- ретроспектива спринта.

Давайте разберем их подробнее.

Все усилия – для удобства внесения изменений, причем с возможностью получить готовый продукт по окончании каждой итерации (цикла). А еще для прозрачности процессов, поднятия морального духа команды, мотивации и, в конечном счете, счастья своих сотрудников, ведь счастливые люди работают намного эффективнее.

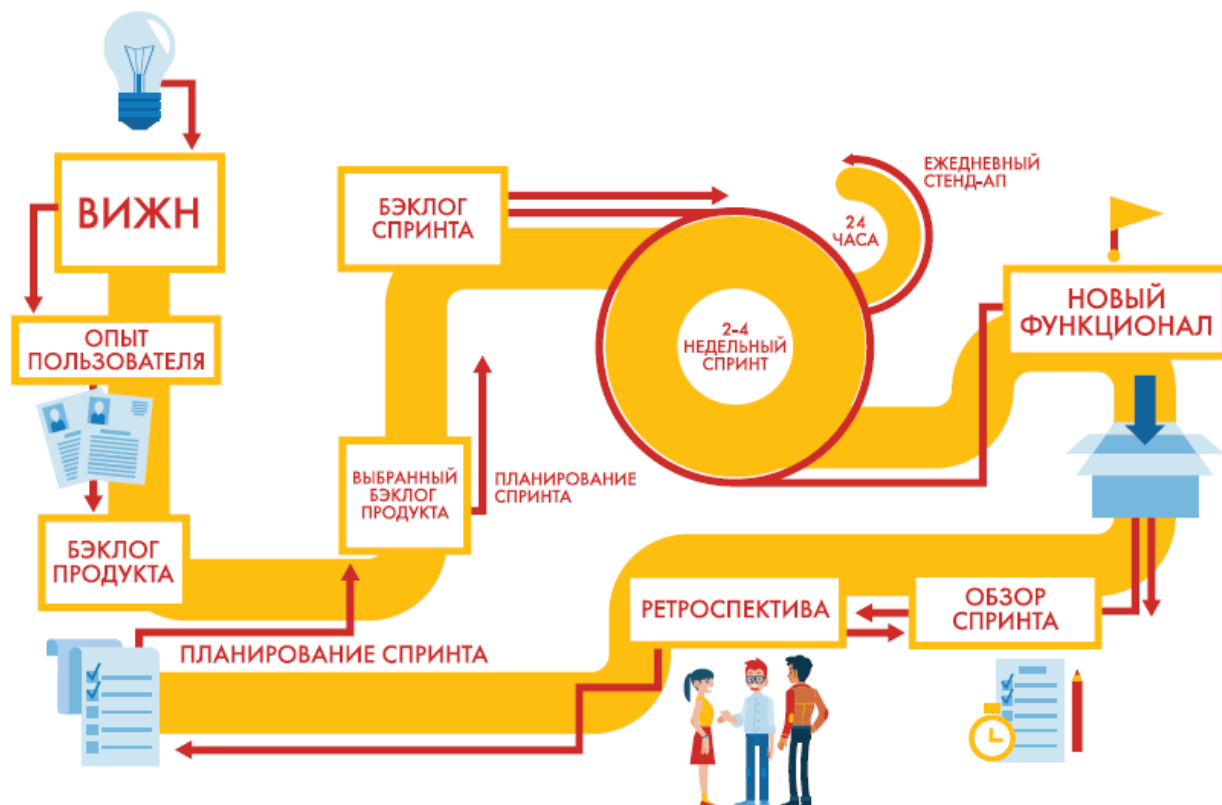


Рис. 11. Схема Scrum

Если коротко, на практике все это выглядит так.

1. Прежде всего назначается **ВЛАДЕЛЕЦ ПРОДУКТА (PRODUCT OWNER)**. Это человек, который лучше всех знает, что за продукт вы делаете, и определяет его видение. Чаще всего именно ему принадлежит идея, и именно он отвечает за бэклог^[38]. В реальных условиях роль этого человека во многом схожа с ролью продюсера проекта.

2. После того как команда собрана, следует выбрать **СКРАМ-МАСТЕРА (SCRUM MASTER)**. Это должен быть самый организованный человек на проекте, способный следить за эффективной работой всех участников. Он не обязательно хороший управленец; главная его задача – помогать всем соблюдать методы Scrum, приоритезировать задачи и улаживать кризис. Можно не назначать его специально: подходящий на эту роль человек сам, скорее всего, не выдержит беспорядка и предложит свою кандидатуру.

Если проект большой и у него есть проектный менеджер, Scrum-мастер чаще всего получает задания от него, после чего самостоятельно выстраивает процессы внутри конкретной команды (например, тестировщиков). Для небольших проектов ПМ чаще всего и становится Scrum-мастером.

Чтобы не возникало конфликта интересов, один человек не должен выполнять роль и Scrum Master, и Product Owner – они должны дополнять друг друга. Кроме этих двоих, все занимаются своей обычной работой.

3. Далее **СОЗДАЕМ БЭКЛОГ** – хранилище всех идей, связанных с проектом. Во многом Scrum – про демократию, так что обычно любой участник команды может внести свою идею для дальнейшего ее обсуждения с командой. К тому же, когда каждый может высказаться, уменьшается риск пропустить эффективное решение.

Бэклог постоянно дополняется новыми инициативами и идеями, так что ситуация, когда ваша игра давно в сторах (магазинах), хотя в бэклоге полно невыполненных задач, встречается нередко.

4. Потом команда собирается, чтобы обсудить и **ОЦЕНИТЬ ЗАДАЧИ**. Молодым командам сложно сразу правильно определить, сколько времени и ресурсов уйдет на то или иное действие. Определите хотя бы два параметра: длительность (насколько задача трудоемкая) и риски (что может пойти не так).

В игровых студиях, работающих по Agile, исполнители часто сами оценивают поставленные задачи. Иногда эту функцию берет на себя Scrum-мастер, например если сотрудник – новичок или ранее он уже оценивал свои задачи неправильно.

5. Далее **ПЛАНИРУЕМ СПРИНТ**. Мы просмотрели, оценили наши задачи и убедились, что можем уложиться в сроки спринта (это может быть неделя, две недели, три недели, месяц).

Спринт – это итерация. Он жестко фиксирован по времени, и каждая команда определяет для себя оптимальный срок спринта. Чем он короче, тем более гибким становится процесс разработки, меньше ресурсов тратится на неправильные задачи, чаще

демонстрируется результат. Однако есть риск много времени и сил тратить на презентации продукта и совещания.

Современный подход говорит о том, что директивно раздавать задачи – не всегда эффективно. Если никто в команде не хочет браться за какую-то задачу, это повод обсудить, насколько проекту вообще она нужна. Это верно для небольших команд. Для крупных комплексных проектов, где все фичи взаимосвязаны (ММО, МОВА и пр.), такое решение может мгновенно привести к коллапсу, так что следует быть осторожным. Однозначных ответов здесь нет, все зависит от конкретной команды и проекта.

6. **ЕЖЕДНЕВНЫЙ СТЕНД-АП** позволяет всем членам команды знать, чем заняты остальные. Scrum должен обеспечивать прозрачность процессов и результатов, так что обычно с такой встречи и начинается рабочий день. Хороший стенд-ап длится недолго. Достаточно, чтобы каждый за пару минут обозначил, каких результатов добился вчера, какие у него планы на сегодня и есть ли какие-то проблемы, мешающие выполнению задачи. Возможные трудности не должны решаться здесь и сейчас, лучше назначить отдельное обсуждение, а на следующем собрании обозначить, что было сделано, чтобы исправить положение.

7. По пятницам обычно команда собирается для **ОБЗОРА СПРИНТА (ДЕМО)**: обсуждаем прошедшую неделю, удалось ли воплотить в жизнь все запланированное, показываем заказчику результаты. Хорошей практикой считается, когда демо проводят не одни и те же люди, это дает разным сотрудникам ощущение причастности к проекту.

8. **РЕТРОСПЕКТИВУ СПРИНТА** лучше проводить также в конце недели, но отдельно от демо. Этот этап – позитивная точка окончания рабочей недели. Здесь не обсуждают задачи (мы уже все обсудили во время демо); это время обсуждения проекта без давления, с целью укрепления командного духа. Правильное окончание ретроспективы: «Ребята, всем спасибо, на сегодня работать закончили, на кухне вас ждет пицца».

Недельная итерация завершилась, можем двигаться к пункту 5 и планировать следующий спринт.

Пункты 5–8 идут циклично. Так что логично планировать спринт в один и тот же день, например в понедельник, чтобы поставить задачи на неделю.

Еще бывает практика больших собраний гейм-дизайнеров, где каждый может высказать свое мнение, идет ли проект в правильном направлении. Например, кто-то в команде может считать, что мы выбрали неправильный арт-стиль и наша игра только выиграет, если мы сменим сеттинг киберпанка на фэнтези. Можно поручить арт-директору еще раз провести исследование, и, даже если решение не изменится, человек будет видеть, что к нему прислушались. Такие собрания особенно эффективно проводить перед планированием.

Зачастую административные процессы (зарплаты, часы работы и пр.) устанавливаются руководителем для всей компании, продуктовые же – можно менять, исходя из команды и проекта: это нормально, если проектный менеджер предпочитает работать, например, в Trello, а лид программистов любит развешивать на доске бумажки. Хотя руководители некоторых игровых компаний настаивают на одинаковых методологиях и подходах для всех проектов студии.

Проектное управление на этапе препродакшен

Классическое проектное управление помогает компаниям укладываться в сроки, выполняя обязательства, а инди-разработчикам дает возможность построить процессы таким образом, чтобы хотя бы довести проект до релиза.

Цель предлагаемых нами моделей – выпустить качественный проект в срок, потратив определенные, в том числе денежные, ресурсы. Поэтому, если вы инди-одиночка, который делает игру для себя и рискует только тратой собственного времени (что, конечно, тоже нежелательно), тщательная подготовка ко всем этапам может быть необязательной. Если что-то пойдет не так, вы просто поработаете над игрой лишний месяц/год. Если вы при этом учитесь или работаете, как обычно, и уверены, что страсть доделать игру не пропадет, то в общем-то в этом нет ничего страшного. Если же время, которое вы тратите на создание игры, не позволяет полноценно зарабатывать, а, как говорится, надо кормить семью, риски многократно возрастают.

Игровые компании в свою очередь рискуют миллионами долларов, и для них правильное построение процессов критично: даже один убыточный проект – это риск закрытия студии навсегда.

Без предварительной документации довольно трудно понять, какие сотрудники на каком этапе разработки могут понадобиться. Ваша идея должна обрести новую форму, которая поможет ответить на этот вопрос. У проекта уже есть концепт, вижн, теперь предстоит еще немного углубиться в детали и составить **FEATURE-ЛИСТ**.

Это краткое описание всех фичей, из которых состоит игра, например строительство базы, клановые бои, система повышений и т. д. Также этот документ – анализ всех действий с точки зрения приоритетов и временных затрат. Он состоит из списка задач, категории, к которой можно отнести задачу (программный код, графика и т. д.), и времени на реализацию. Можно просто оценить

задачи одной строкой по времени, а можно разбить по этапам (препродакшен, продакшен или релиз).

Обычно такой документ составляется в два этапа: первый раз вы встречаетесь всей командой и просто составляете полный список фичей, затем каждый ответственный заполняет свою часть. Вторая встреча посвящена согласованию этих оценок, после чего можно приступать к планомерной работе. Грамотно составленный feature-лист может сэкономить в будущем уйму времени.

Многие разработчики, особенно начинающие, игнорируют этот документ, так как неопытным специалистам составлять его нелегко. В этом случае можно обойтись просто оценкой бэклога, о которой мы писали выше, когда говорили о Scrum. Однако помните, что без feature-листа вы сможете оценить общее время разработки игры только очень приблизительно. Именно на основании этого документа вы (ваш продюсер или ПМ) будете ставить конкретные сроки для своих задач.

Мобильная аналитика					
# №	Задача		Описание	Приоритет	Дедлайн
1	Административные вопросы		Описать задачи по прогнозированию и отслеживанию мобильного трафика. Усилить команду человеком, имеющим сильную экспертизу в мобильной аналитике	900	20-Jun
2		Найти ведущего мобильного аналитика	Все организационные вопросы, решение которых необходимо для выполнения задачи по мобильной аналитике: кадровые вопросы, регламенты взаимодействия, разгрузка от текучки и т.д.	850	1-Jul
3	Перенять возможный опыт у Nexters и Webgames		Понять, какие вопросы мы хотим задать. Договориться с Максом о поездке к коллегам для обмена опытом	850	13-Jun
4		Выбрать систему (платформу) аналитики	понять, что нужно нам. Зафиксировать процедуру актуализации этой информации, чтобы оставаться в тренде	800	20-Jun
5		Подключить мобильную аналитику к нашей системе	Реализовать техническое хранение данных, выгрузку данных из системы в нашу базу, их обработку и передачу в Табло	600	1-Jul
6		Интегрировать мобильную аналитику в Империял	Подключить выбранную аналитическую платформу к Империялу, убедиться в корректности сбора всей необходимой статистики	500	18-Jul
7	Расчёт LTV мобильного трафика		Уметь считать LTV по каналам и сравнивать его с CPI (без учёта кросс-платформенности)	900	1-Aug
8		Учёт кросс-платформенного трафика	Сегментировать аудиторию по каналам привлечения в условиях отсутствия рефов, считать маркетинговый ROI и окупаемость группы каналов (методов привлечения)	850	5-Sep
9		Прогнозирование годовой выручки	Аналитика для бизнеса: прогнозы выручки на год, срока существования проекта и т.д.	500	19-Sep

Дедлайн	Осталось дней	Кросс-платформенные системы	Декларируют поддержку multy-channel engagement
8/15/2016	-1136	deltaNDA devtodev GameAnalytics Localytics Mixpanel Flurry	Localytics
Всего больших задач	11		

Задача	Версия	Отдел	Статус	Приоритет
Мокапы и описание кланов	25 Сентябрь	ГД		1000
Нарисовать интерфейсы клана	25 Сентябрь	Арт	done	950
Проффилирование матч-3 (1-я итерация)	25 Сентябрь	Код		900
Оптимизация матч-3 (1-я итерация)	25 Сентябрь	Код		850
Проффилирование фермы (1-я итерация)	25 Сентябрь	Код		800
Оптимизация фермы (1-я итерация)	9 Октябрь	Код		750
Определение баланса для 52-162 миссий матч-3	25 Сентябрь	ГД	done	750
Баланс и заполнение orders.json для игроков 52+ миссий матч-3	25 Сентябрь	ГД		700
Определение перечня наград миссий 52-162	9 Октябрь	ГД		650
Описание и мокапы театра	25 Сентябрь	ГД	done	600
Корректировка mapObjects.xml и resource.xml для уровней 10+	25 Сентябрь	ГД	done	550
Сделать вместо Нарека иконку магазина	9 Октябрь	Код		500
Сделать кликабельным кораблик	9 Октябрь	Код		500
Интеграция анимации выросших культур для сбора	9 Октябрь	Код		450
Сделать Нарека кликабельным	9 Октябрь	Код		400
не блокировать скролл баблами над квесттиверами	9 Октябрь	Код		400
нарисовать здание театр	25 Сентябрь	Арт	done	375
нарисовать здание "дружбы"	23 Октябрь	Арт		350
нарисовать здание клана	9 Октябрь	Арт		325
нарисовать элементы генерации гербов и флагов (5 фонов, 5 узоров, 10 эмблем)	9 Октябрь	Арт		300
нарисовать первую порцию смайликов для чата (6 штук)	9 Октябрь	Арт		275
перерисовать весь декор, похожий на мусор (кусты, статуи)	25 Сентябрь	Арт	done	250
нарисовать графику по доп механикам матч-3 (орехи, полочки, пёд)	25 Сентябрь	Арт	done	225
нарисовать здание крафта бустеров (лабораторию алхимика)	23 Октябрь	Арт		200
описание и мокапы друзей	9 Октябрь	ГД		175
работа с конфигами по новым механикам (кланы, театр и т. д.)	9 Октябрь	ГД		150
механики матч-3: Орехи, Другой тип пробелов, Течение, Полочки, Замороженные фишки	9 Октябрь	Код		125
нарисовать ресурсы "золотой порошок", "копчан молний" и "огненная соль", необходимые для крафта бустеров	23 Октябрь	Арт		100
создание 75 новых миссий	9 Октябрь	ЛД		75

Рис. 12. Пример feature-листа

Чтобы не терять понимания, когда мы вообще сможем запустить проект, лучше заранее составить майлстоуны^[39]. Допустим, мы насчитали 10 фичей, по одному месяцу на реализацию каждой. Так что мы надеемся завершить проект за десять месяцев. Но что должно быть готово, например, на пятом месяце разработки? Укладываемся ли мы в сроки? Если мы хотим сделать клановые войны, логично, что перед этим надо завершить создание самих кланов. Одни фичи могут тянуть за собой другие. Чтобы не блуждать во тьме, мы разбиваем список задач на отдельные большие блоки с конкретными сроками. Для людей, работающих за деньги, это

критично, ведь ресурсы нужно распределить так, чтобы их хватило до конца проекта.

Майлстоуны делят на более мелкие итерации, короткие промежутки времени, за которые необходимо успеть сделать определенный набор фичей. Если задачи комплексные, лучше разбивать их на более мелкие.

Обычно в feature-листе вы можете найти ссылку на отдельные части ГДД (гейм-дизайн-документа) с более подробным описанием каждой задачи.

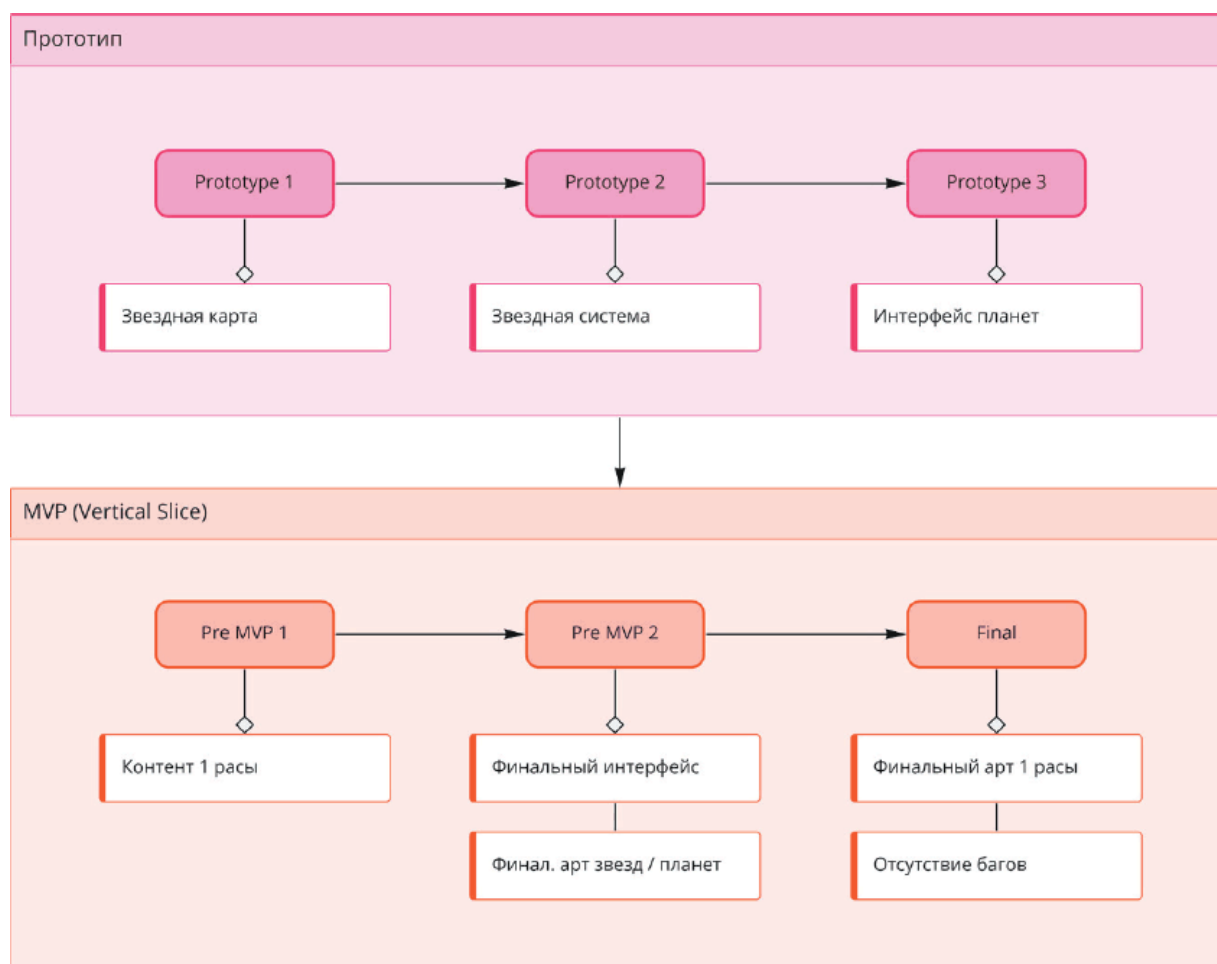


Рис. 13. Майлстоун проекта

ФИЧЕКРИП^[40] И ФИЧЕКАТ^[41]

Обычно жанр определяет некоторое количество игровых фичей, ожидаемых игроками. Гейм-дизайнеры стараются добавить также несколько свежих фичей, чтобы выделить свой проект среди конкурентов и чтобы игра стала интереснее. Принимая решение, гейм-дизайнеры всегда должны проанализировать три вещи: насколько фиша полезная, насколько рискованная и насколько дорогая (сколько человеко-часов нужно для ее реализации).

Ошибки в оценках чаще всего происходят из-за того, что геймдизайнер не смог объяснить исполнителям, что именно представляет собой фиша, над которой им предстоит работать. Особенно это актуально для экспериментальных проектов и команд новичков. Если на проекте нет ни одного опытного лида, который в состоянии хотя бы примерно оценить время реализации, можно попробовать дать исполнителям пробные задачи и по результатам работы сделать соответствующие выводы, а можно обратиться за внешней консультацией.

За фишу отвечают как минимум три человека: гейм-дизайнер, который ее придумал и описал, исполнитель, реализовавший ее в срок, и специалист по тестированию, который проверил, что все работает именно так, как планировал гейм-дизайнер. Последний должен составить ее полное описание, а исполнители и продюсер – дать оценки, насколько фиша важная и рискованная. После этого уже можно приступать к составлению плана работы. Однако не всегда мы можем придерживаться планов, самые разные факторы способны принудить срочно поменять, добавить или убрать ту или иную фишу.

Зачастую фичекрип – это не проблема добавления новых фичей, а проблема того, что текущие фиши влекут за собой новые изменения. Здесь не обойтись без внутренней экспертизы, и, к сожалению, не всегда исполнители могут адекватно оценить

стоимость добавления того или иного игрового элемента. По хорошему исполнители должны доверять решениям гейм-дизайнера. В свою очередь он, прежде чем предлагать нововведения, должен задуматься, какие еще фишки придется отменить или отложить, чтобы реализовать свою, и какими будут последствия.

Лучше еще в самом начале разработки определиться с тем, какие фишки нельзя «вырезать» ни в коем случае, какие – только в случае крайней необходимости, а чем можно пожертвовать. Фичекат может быть болезненным, но чаще всего это полезный процесс, помогающий команде сконцентрироваться на действительно важных вещах и не затягивать разработку.

Иногда, вместо того чтобы просто вырезать что-то, разработчики принимают решение разбить фичу на части и выдавать игрокам контент постепенно. Это позволяет выделить больше времени на тестирование и успеть внести изменения, если что-то пошло не так. Если на каждом этапе такая фича вносит в геймплей что-то новое, игроки будут с нетерпением ждать ее развития. Но есть риск, что получится наоборот. Если, например, игрок видит на карте место для ловли рыбы, но при этом в игре все еще не реализована механика рыбалки, ему становится очевидно, что он купил недоделанную игру, и это может вызвать негативную реакцию.

В течение спринта на крупных проектах выполняются сотни задач, и один человек просто не сможет вникнуть в каждую, чтобы правильно ее приоритезировать. Поэтому очень важно, чтобы заказчик самостоятельно понимал ценность предлагаемой фичи, а исполнитель – риски ее реализации.

В команде должно быть четкое понимание, какие новые идеи превращаются в задачи. Их может утверждать продюсер или демократическое голосование, но важно избежать ситуации, когда, с одной стороны, приоритетные фишки не делаются, а с другой – сотрудники постоянно отвлекаются на новые задачи, пока не выполнены текущие.

Если каждый выбирает только то, что ему нравится, есть немаленькая вероятность, что проект не будет доведен до конца. Для инди-команд такая демократия может работать, если люди «в доле». Если проект не завершится или не получится, они потеряют не только время, но и деньги. Поэтому важно, чтобы в команде был визионер – человек, понимающий ситуацию в целом, – и чтобы к его мнению прислушивались.

Большие проекты всегда предполагают сложный менеджмент. С одной стороны, важно не подавить инициативу сотрудников, с другой – объяснить, что не все нововведения хороши и иногда для проекта лучше следовать заранее установленному плану. Здесь очень важна работа креативного директора, или Scrum-мастера, или лида направления, обучающего людей самостоятельно принимать взвешенные решения.

ДОКУМЕНТАЦИЯ: БЮДЖЕТ ПРОЕКТА И БИЗНЕС-ПЛАН ПРОЕКТА

Бюджет проекта – это план затрат, необходимых для его исполнения. Сюда включают заработную плату сотрудников, стоимость оборудования, закупку материалов, программного обеспечения, ассетов, налоги и пр. В игровой индустрии обычно он состоит из двух частей: планируемые расходы и прогнозы доходов. К сожалению, без грамотно составленного feature-листа сделать адекватный план бюджета и бизнес-план невозможно. А это влечет за собой большую проблему: если бюджет будет сильно завышен, то издатель/инвестор не захочет с ним работать, если же вы ошибетесь в сторону уменьшения, не исключена ситуация, когда вы подпишете договор на определенную сумму, а деньги закончатся задолго до завершения проекта.

План бюджета – это обычно именно внутренний документ, по которому фактически работает команда. А бизнес-план

предназначен для демонстрации планов и прогнозов инвесторам/издателям.

Документация и инструменты управления помогают понять, в какой последовательности реализовывать фичи, где может не хватить ресурсов и времени, где проект может «просесть». Задачи разбиваются на более мелкие и структурируются по времени и приоритету.

Нельзя не сказать о том, что четкое планирование больше подходит для крупных профессиональных команд, потому что имеет те же недостатки, что и Waterfall: любые планы имеют тенденцию расходиться с реальностью, особенно когда вы делаете что-то впервые. Идеи, которые вы собираете в бэклоге, – это прежде всего цели, и для многих команд эффективнее дать исполнителям определенную свободу на пути к их достижению. Главное, чтобы эти задачи были реальными и выполнимыми в указанные сроки.

Выбор инструментов не в последнюю очередь зависит и от самого проекта. Для клиентских однопользовательских игр дорожные карты (roadmap) и майлстоуны (milestones) работают хорошо. Если компания заключила договор с инвестором или издателем, в нем так или иначе будет прописано, что и в какой срок должно быть готово и что произойдет в случае неисполнения. В такой ситуации не составлять четкие графики и планы очень рискованно. Небольшие мобильные игры с понятными механиками и опытной командой тоже пользуются всеми этими инструментами, так как производство зачастую поставлено на поток.

Для неопытных команд или экспериментальных проектов все, конечно, несколько сложнее. Много может меняться на ходу, ведь риски и приоритеты могут быть изначально определены неверно.

И конечно, каждая команда индивидуальна; выбирайте или придумывайте что-то свое, главное – результат.

Выбор игрового движка

Если с планированием более или менее определились, пора переходить к важному техническому решению – выбору игрового движка.

Термины «игровой движок» и «игра» тесно переплетены, разработчики могут по-разному понимать границы между ними. Мы будем рассматривать игровой движок как базовое программное обеспечение, пригодное для повторного применения и расширения, служащее основой для создания различных игр.

Игровые движки в современном понимании зародились в середине 90-х. В то время инди-разработчики как таковой практически не было, созданием игр занимались более или менее профессиональные компании – хотя, конечно, по современным меркам многие из них были инди. Каждый движок делался под специфический жанр или игру.

Если вы делаете игры с похожими механиками, каждый раз создавать движок с самого начала неэффективно. Поэтому было решено отделить самую базовую игровую логику и просто дополнять ее необходимыми для каждой конкретной игры элементами.

От текстовых адвенчур (*Adventure Construction Set*, 1984, или *Graphic Adventure Creator*, 1985, и др.) разработчики игр постепенно перешли к созданию шутеров. Именно после появления *Doom* (созданного на легендарной серии движков *id Tech* от *id Software*, развитие которой продолжалось более 25 лет – от *id Tech 1* в *Doom*, 1993, до 7-й версии, использованной для *Doom Eternal*) термин «игровой движок» приобрел современное значение.

Теперь игровая логика и механики отошли на второй план; новые технологии ценились за честную картинку, которой раньше не было, физику и возможности рендера (визуализации). Удачное сочетание центральных компонентов игры (подсистем графики, звука, расчета столкновения объектов и других) позволило создавать новые

проекты на базе готового игрового движка, что сильно сократило время разработки.

Позже появились движки Unreal Engine (1998, первая разработанная игра – шутер от первого лица *Unreal*) и CryEngine (2002, первоначально используемый для создания *Far Cry*), позволяющие воспроизводить топовую графику. Они были довольно сложными и стоили сотни тысяч долларов, так что пользоваться ими могли в основном только сотрудники крупных игровых компаний.

В то время самостоятельно издавать игры было очень тяжело: следовало договориться с магазинами о размещении дисков, а прилавков обычного магазина, в отличие, например, от Steam, физически не может вместить все выпускаемые игры. По этой же причине разрабатывать нишевые игры с небольшой аудиторией поклонников было сложно.

Только с появлением цифровой дистрибуции и развитием рынка мобильных игр разработчики получили возможность знакомить со своими играми широкую аудиторию. Теперь современные движки стали удобным инструментом создания самых разных игр. Сначала Unity, а позже CryEngine и Unreal Engine стали условно бесплатными, что сделало эти движки доступными практически для любых разработчиков.

Сегодня в базе игровой движок дает «строительные кубики», из которых собирается игра, а также большой выбор инструментов, облегчающих жизнь разработчика. Вам не придется изучать множество материалов по программированию, однако совсем без таких навыков гейм-дизайнеру будет непросто. Зная, как работать хотя бы с одним языком программирования, вы относительно легко сможете переключиться на другой. Если писать код вы не хотите, все равно вам придется составлять техническое задание своим программистам, а значит, вы должны хотя бы в общих чертах понимать, какие методы они могут использовать для решения различных задач. В общем, в этой области знания нужно будет подтянуть. Существует очень много вариантов для визуального

программирования, облегчающих жизнь разработчика. Если сегодня вы можете работать только, например, с блупринтами^[42] Unreal Engine или в GameMaker^[43], это лучше, чем ничего.

ЧТО ДАЕТ ИГРОВОЙ ДВИЖОК

- Главное достоинство любого движка – эффективный рендеринг графики, позволяющий сократить время разработки.
- Движок задает базовую физику, например гравитацию, так что не нужно самому вспоминать законы Ньютона; у предметов уже есть параметры того, как они должны лежать, падать и т. п. Системы моделирования взаимодействия игровых объектов помогают правильно определить, попала ли пуля во врага, если это шутер, или как колеса автомобиля сцепляются с дорогой в гонках.
- Также движок часто дает сетевой протокол. Если мы делаем онлайн-игру, где один игрок стреляет в другого, информация об этом уйдет на сервер, где результат будет посчитан, обработан и отправлен игрокам.
- Движок дает систему искусственного интеллекта. Например, можно определить, как монстры будут перемещаться, бегать группами, атаковать игрока и т. д.
- Сегодня движки ценятся за кросс-платформенность, то есть за возможность разрабатывать с их помощью игры разных жанров для разных платформ (PC, PlayStation, Xbox, Nintendo Switch, iOS, Android пр.). Чем больше устройств смогут поддерживать вашу игру, тем больше людей будут иметь возможность ее оценить.
- Как правило, в движке есть средства для создания игрового контента. Вы найдете множество инструментов для моделлеров, аниматоров, художников по окружению, тестировщиков и других членов команды.
- Некоторые движки дают средства для разработки игровой логики. Если вы хотите, чтобы, открывая какую-то дверь, игрок

сталкивался с монстром, достаточно прицепить к двери триггер события «открытие», а к монстру – действие «появиться». Гейм-дизайнер без специальных знаний программирования с помощью понятного интерфейса сможет собрать игровую логику. Хотя есть движки, требующие написания кода для добавления таких триггеров.

- Часто движки дают средства оптимизации и отладки, возможность проверить проект на ошибки и увидеть, на какие части игровой логики или рендеринга тратится больше всего времени.

- Важное преимущество готовых движков – комьюнити. Вы всегда сможете узнать у техподдержки или у других пользователей, как сделать то, что вам необходимо. Чем больше комьюнити, тем проще найти готовое решение.

- Отдельно нужно отметить наличие у движков магазинов с платными и бесплатными готовыми решениями: системой оплаты, библиотеками для работы с физикой и др. Например, Unreal Engine часто дает пользователям бесплатные функции, будь то система красивых аутлайнов^[44], физика пробивания пуль различных материалов или генерация правдоподобного леса. Разработчики Unity, имея примеры готовых игр в магазине, выкладывают новые наработки под свободными лицензиями.

КАК ВЫБРАТЬ ИГРОВОЙ ДВИЖОК

Выбор игрового движка – это принципиальное техническое решение, его смена в процессе разработки практически невозможна. Обычно решиться на такой радикальный шаг может подтолкнуть только какая-то глобальная катастрофа, например понимание, что изначально принято неверное решение о выборе движка и игру просто не получится доделать на текущем. Переделывать придется очень многое или почти все, поэтому стоит со всей ответственностью подойти к этому вопросу.

Конечно, обладая навыками программирования, вы можете сами написать код движка. Крупные компании, работающие над экспериментальными решениями и имеющие специалистов должного уровня, нередко выбирают такой путь. Но нужно помнить о том, что это очень дорого. Многие движки, особенно создаваемые для сложных комплексных игр, разрабатывались с бюджетами в 10–20 миллионов долларов, то есть стоили как полноценная AAA-игра^[45]. Разработчики новых движков могут рассматривать их как вложение в будущее – для новых игр по франшизе или для продажи.

Для небольших компаний или инди-команд решение самостоятельно создавать движок не имеет экономических преимуществ, но может быть оправданно в случае необходимости каких-то необычных решений, которые невозможно получить, используя готовый вариант. Это могут быть вещи, связанные, например, с воксельной графикой, где важно передать разрушаемость предметов. Для игр типа *Minecraft* характерно использование собственных движков, так как графика там достаточно примитивная, но предметы должны уметь правильно разрушаться. Или же вы хотите сделать игру с необычным базовым типом геймплея: например, головоломку, где нужно на призмах раскладывать световые лучи в радугу. Если готовые движки не смогут предложить вам готовых решений для передачи оптики, придется создавать эту часть игры своими силами.

Еще один вариант – не делать движок с нуля, а взять наработки опенсорсных (от англ. open-source), то есть выложенных в открытый доступ, движков. Исходный код таких движков доступен всем, и его можно править под свои нужды. Естественно, вам понадобятся опытные программисты, чтобы справиться с такой задачей. Многие популярные современные движки имеют гибкие инструменты для расширения и кастомизации (разной степени сложности и удобства). «Прикрутить» уникальную механику к готовому движку намного

легче, чем с нуля создавать свой полноценный движок с аналогичными возможностями.

Если же вы планируете создать игру с более или менее типовым геймплеем, готовые движки сильно упростят задачу. У них есть документация, где всегда можно прочесть, что он умеет или не умеет делать. Самый простой способ узнать, реализует ли выбранный движок ту или иную игровую механику, это изучить его описание или же просто задать вопрос в поисковиках.

Готовые игровые движки можно разделить по двум признакам. Во-первых, по игровой платформе: движки могут быть кросс-платформенными или же подходить только для создания игры под конкретную платформу (мобильные, ПК-игры и т. д.).

Во-вторых, важен жанр создаваемой игры. Существуют игровые движки, так скажем, широкого профиля, поддерживающие игры разных жанров, но есть и специализированные движки, дающие инструментарий для создания, например, визуальных новелл (Ren'Py). Движок, спроектированный для гонок, будет существенно отличаться от движка для MMORPG или стратегии в реальном времени. Но есть и общие вещи: например, трехмерные игры любого жанра требуют системы 3D-рендеринга, взаимодействия с геймпадом, клавиатурой и/или мышью, звукового сопровождения и так далее.

Еще один важный фактор при выборе движка – опыт участников проекта, в том числе языки программирования, которыми они владеют.

МНОГОПРОФИЛЬНЫЕ ДВИЖКИ UNITY И UNREAL ENGINE

Самые популярные игровые движки – это Unreal Engine и Unity.

UNITY развивается с 2005 года. На Unity разработаны тысячи приложений и игр разных жанров на более чем 25 платформах. Его любят как крупные студии, так и инди-разработчики.

Изначально Unity создавался для компьютеров Mac, в следующих версиях добавлялись новые платформы: Windows, iPhone, Android, Xbox, Playstation и другие.

Для написания скриптов движок использует язык программирования C#, считающийся несложным в изучении и работе. В редакторе Unity простой интерфейс, что позволяет легко производить отладку игры. Главными преимуществами Unity считают кросс-платформенность и наличие визуальной среды разработки.

Если выбирать между Unreal Engine и Unity, можно отметить, что последний иногда удобнее для начинающих программистов, он предлагает множество готовых решений и для создания простых игр обычно выбирают именно его. Сообщество Unity считается самым большим (ведь он стал первым бесплатным движком для инди-разработчиков), поэтому, если возникнут какие-то вопросы, вы всегда сможете рассчитывать на консультацию. Однако, если с программированием отношения сложные, Unreal Engine может кому-то показаться более дружелюбным.

Работая с любым движком, сложно добиться AAA-графики в динамике, особенно для консольной игры. Планка качества графики и спецэффектов непрерывно растет, организация совместной работы над огромным количеством внутриигровых объектов усложняется. За последние годы разработчики Unity проделали большую работу для повышения уровня графики, так что сегодня создавать красивые и качественные игры можно на обоих движках.

Большие открытые миры – еще одна сложная задача. Поэтому многие известные игры (*Grand Theft Auto*, *Red Dead Redemption*, *Metal Gear Solid 5*, *Horizon: Zero Dawn* и другие) сделаны на собственных движках, заточенных под эту техническую задачу. Многие подсистемы Unity не рассчитаны на создание чего-то подобного – к примеру, на организацию большого количества сцен и объектов. В ряде случаев они накладывают серьезные ограничения на размер внутриигрового мира, но их можно обойти с помощью уловок гейм-

дизайна. Unity постоянно работает над новыми технологиями для решения таких задач.

Раньше у Unity были сложности с реализацией защиты от читов^[46]. Считалось, что его игровой клиент легче вскрывается, а значит, для игр, где читерство могло стать проблемой, разработчики задумывались о другом решении. Сегодня при прочих равных взлом игры, разработанной на Unity, не будет проще; компания постоянно работает над укреплением надежности клиент-серверной архитектуры и шифрованием внутриигровых данных. Студии, занимающиеся мультиплеерными проектами, нередко прибегают к сторонним решениям для защиты от читов, например PunkBuster.

Плюсом Unity является модульная система, что позволяет подключать нужные модули физики, рендеринга, аналитики, монетизации и т. д., не выходя из движка.

Unity – один из самых универсальных движков, с его помощью можно создавать игры любого жанра. Он удобен для разработки, например, CRPG с видом три четверти или изометрическим видом (*Pathfinder: Kingmaker*, *Pillars of Eternity* и т. п.). Также хорошо использовать Unity для survival-игр (игр про выживание), где обычно не очень большая карта, простая графика и умеренная динамика. Удачные примеры игр на Unity: *Hearthstone*, *Ori and the Blind Forest*, *Escape from Tarkov*; мобильные проекты – *Pokemon Go*, *Super Mario Run* и другие.

UNREAL ENGINE выпустила в 1998 году компания Epic Games. Это один из первых универсальных движков, совмещающих физику, рендеринг, искусственный интеллект и готовую среду разработки. Изначально он создавался для работы над шутерами от первого лица, однако последующие версии применяются для игр самых разных жанров. Unreal Engine всегда уделял много внимания впечатляющей картинке, им пользуются и в кинематографе; сегодня он предоставляет продвинутый инструментарий для создания фотореалистичной 3D-графики в реальном времени. Компания Epic Games и сама делает игры на своем движке, в том числе одну из

самых зарабатывающих игр в мире, реализованную на всех игровых платформах, – *Fortnite*.

Преимуществом Unreal Engine является мощная система блупринтов, с помощью которой легко и удобно прорабатывать игровую логику. Блупринты – это скриптовая система, которая представляет собой визуальный интерфейс для создания элементов геймплея. Она позволяет гейм-дизайнерам, вообще не знающим кода, использовать почти полный потенциал программирования. Если вы хотите создавать контент (интерфейс, геймплей, игровые уровни и т. д.) самостоятельно, удобно работать с Unreal Engine, хотя для некоторых вещей классический код остается в приоритете.

А вот если вы планируете использовать много готовых ассетов, это может быть плюсом к выбору Unity, имеющему огромный магазин. Разработчики Unity создали также собственный инструмент визуального скриптинга – Bolt, позволяющий реализовывать игровую логику без ручного ввода программного кода.

Unreal Engine имеет много готовых решений по графике. Он хорошо поддерживает технологии RTX^[47], регулярно выпускает обновления, дающие новые возможности создания впечатляющей картинки. Удобный редактор позволяет прописать игровую логику без специальных навыков программирования, а реалистичная графика, игра света и тени, звуки создают необходимую атмосферу.

Вы всегда можете ознакомиться с актуальными условиями использования движков в открытых источниках и решить, какой из них выгоднее использовать для вашей игры.

Если к открытому миру без загрузок добавляется AAA-графика и большое количество взаимодействия с миром, можно рассмотреть движок **CRYENGINE**: он как раз стоит на стыке между специализированными и универсальными движками. *Kingdom Come: Deliverance*, например, сделан именно на нем. У CryEngine тоже есть функционал для визуального скриптования Flow Graph. У этого движка больше требований, поэтому на слабом железе такую игру не запустить, что сразу несколько ограничивает вашу аудиторию. Но

опытные команды с его помощью могут создавать по-настоящему масштабные игровые проекты.

СПЕЦИАЛИЗИРОВАННЫХ ДВИЖКОВ тоже довольно много. Они реализуют какой-то конкретный нестандартный запрос лучше, чем универсальные движки. Обычно такой движок умеет только что-то одно, зато очень хорошо, в то время как многопрофильные движки умеют почти все, но на среднем уровне. К примеру, Buildbox прекрасно справляется с задачей создания игр на простых механиках без навыков программирования.

Существуют также **ДВИЖКИ ДЛЯ НАЧИНАЮЩИХ РАЗРАБОТЧИКОВ**, такие как GameMaker: Studio, Stencyl и др. В отличие от Unity, все-таки требующего каких-то знаний о C#, они вообще не предполагают наличия навыков программирования. Это конструктор, где вы можете собрать игру из готовых частей. Создать новую механику на базе такого движка почти невозможно. Но если разработчики хотят сделать игру дешевле и проще, то это отличный вариант: например, *To the Moon* была сделана на базе движка RPG Maker. Если у вас стилизованная графика и нет сложного геймплея, то такие готовые движки – хорошее решение. Еще очень удобно использовать их для создания прототипов или игр, которые делаются в рамках гейм-джемов^[48].

Также нужно упомянуть о **ПРОПРИЕТАРНЫХ (ЗАКРЫТЫХ) ДВИЖКАХ**, доступ к которым любому желающему не предусматривается, – например, Frostbite, на котором разрабатывается *Battlefield* и другие. Они могут быть любой категории: специализирующиеся на одной определенной задаче, общие и т. д. Как правило, такие движки принадлежат какой-то компании, и воспользоваться ими разработчик может только по особым каналам, заключив соответствующее соглашение на использование.

Итак, чтобы сделать выбор, прежде всего нужно отсеять движки, которые не подходят выбранной игровой платформе. Если вы делаете типовой проект, скорее всего, лучшим решением станет

выбор универсального движка, Unity или Unreal. Если вам нужны оригинальные решения или игра специфического жанра, посмотрите более специализированные варианты: например, создавать визуальную новеллу лучше на базе специально разработанных для этого жанра движков. Если в вашей команде нет опытного программиста, стоит обратить внимание на движки для начинающих. Если для проекта нужно много программирования, а ваш программист уверенно владеет только C#, имеет смысл выбрать соответствующий движок.

Далее следует определиться с требованиями к качеству графики. Даже инди-команды, используя, например, Unreal, могут рассчитывать на качественную и красивую картинку при наличии хороших моделей. Помните, что пропасть между «отличной» и «просто хорошей» графикой за последние годы сильно сократилась.

Использовать опыт предшественников – тоже хорошая идея. Если у вашей игры есть референсы, логично посмотреть, на каком движке они были созданы, и проанализировать, почему было принято то или иное решение.

Игровые прототипы

Прототипы – это неконечные игровые продукты, создаваемые для того, чтобы проверить свои идеи и ответить на вопросы. Например:

- работают ли наши гипотезы и основные фичи;
- работают ли они так, как мы запланировали;
- играют ли люди в нашу игру так, как мы запланировали;
- может ли понравиться игра нашей аудитории;
- какие части нашей игры вызовут у нас наибольшие сложности.

Создание прототипов позволяет как можно скорее выявить слабые стороны проекта и устранить их. Прототип, созданный с целью ответить на конкретный вопрос или вызов, – самый полезный. «Интересно ли будет играть в мою игру?» – пример плохого вопроса. Лучше: «Как долго сохраняется интерес к основному геймплею?» или «Сколько анимированных объектов в одной сцене в максимальном качестве поддерживает наш движок?» Версия для среза первого впечатления от игры не сможет ответить на вопрос, не будет ли скучно играть пятнадцатую сессию, поэтому следует четко обозначить цель исследования.

ВИДЫ ИГРОВЫХ ПРОТОТИПОВ

ГЕЙМПЛЕЙНЫМИ ПРОТОТИПАМИ называют версию продукта, в которую так или иначе можно поиграть и проверить игровые механики. Например, как работает боевая система или сколько нужно времени, чтобы пробежать конкретный уровень платформера.

Проверяется либо комбинация более или менее стандартных идей, либо жизнеспособность и последствия инновационных решений. Цель здесь – выявить проблемы, над которыми мы по тем или иным причинам не задумывались.

Ответы нужно получить максимально быстро. Если прототип отвечает на поставленные вопросы, абсолютно неважно, в каком качестве он сделан.

Бывают чисто **ФУНКЦИОНАЛЬНЫЕ ИЛИ МЕХАНИЧЕСКИЕ ПРОТОТИПЫ**, созданные, чтобы проверить гипотезы без каких-либо программ. Это могут быть, например, бумажные прототипы: настольная игра или просто нарезанные листочки бумаги. Таким способом отлично тестируются всевозможные разновидности стратегий и карточных игр.

Если ваша игра про принятие решений и игровую механику, бумажные прототипы – это то, с чего следует начинать. Это экономит кучу времени и сил, ведь выкинуть бумажку и написать новую – намного проще, чем править код.



Рис. 14. Прототипы могут быть даже бумажными!

Этот способ будет вполне эффективным даже для создания прототипов шутеров от первого лица. Если на каждое действие вы

выделите своим игрокам определенное количество секунд (с помощью секундомера, например), то сможете пошагово изучить механики даже для игры в реальном времени: сколько шагов или выстрелов в секунду может сделать игрок, какого размера карта нужна для каждого уровня, какой вид оружия эффективен против монстров с разными характеристиками, сколько патронов и здоровья необходимо и так далее.

Если играется хорошо, стоит попробовать собрать прототип в простой игровой системе. С помощью готового игрового движка, разработка на котором будет наименее затратной, проверяем, насколько интересны придуманные нами игровые механики, полностью пренебрегая графикой. На этом этапе стараемся использовать максимум готовых решений: программы и низкоуровневые движки, которые позволяют быстро скомпоновать из ассетов базовый геймплей. Если игра планируется на Unreal Engine 4, прототип для нее все равно вполне можно собирать с помощью Game Maker и аналогичных программ, главное – быстро, просто и дешево; лучше использовать знакомые инструменты, чтобы не тратить время на изучение новых.

Для создания прототипов можно использовать и чисто графические методы (Flash, Figma и пр.), тогда эта работа ложится на плечи художника-дизайнера. Этот метод подходит для игр, где важно проверить, например, как решения игрока влияют на геймплей (в какую сторону он отправился, какой вариант в ветке диалогов выбрал и прочее).

Ради упрощения создания прототипов допустимо срезать целые пласты геймплея. Для комплексных, сложных игр лучше вообще тестировать отдельный функционал и игровые механики.

Когда вы проверяете какие-то инновационные идеи, которые нельзя найти в Unity или Unreal Engine, задача усложняется. Есть несколько вариантов решения этой проблемы.

Если мы хотим протестировать какую-то комплексную систему, можно попробовать использовать наработки предшественников.

Допустим, вы хотите добавить в игру механику осады замков. Чтобы оценить, впишется ли фича в игру, уже многое должно быть готово (боевая система, прокачка персонажей). Создавать целую игру, чтобы проверить одну фичу, долго и неэффективно. Стоит постараться найти движок наиболее похожей игры и «прикрутить» туда нужный функционал.

Другая ситуация: сложный технический запрос. Например, гейм-дизайнер задумал сделать жидкого персонажа, который должен правдоподобно перетекать, менять форму, иметь определенные способности и адекватно взаимодействовать с более привычными героями. Такого с помощью готовых решений не соберешь, подыскать подходящую программу может быть проблематично. Обычно над такими нетривиальными задачами трудится отдельный программист, задача которого – максимально быстро написать код нужной нам системы. Такой прототип не ответит на вопрос о стабильности решения или совместимости с полноценным движком, просто нужно в сжатые сроки сформировать и оценить задуманный геймплей.

МАТЕМАТИЧЕСКИЕ ПРОТОТИПЫ создают, чтобы убедиться, правильно ли работают математические расчеты. Причем методы могут быть вообще не связаны с реальной игрой, а работающий над ними специалист может обладать знаниями скорее в области математики, нежели гейм-дизайна.

Чтобы создать **АРТ-ПРОТОТИП**, необходимо проанализировать игры, которые мы считаем конкурентами, и оценить стандарты индустрии для выбранного жанра. Причем не только сегодня, но и на момент предполагаемого выхода игры. После этого арт-директор и технический художник (или кто-то, выполняющий их функции) принимают решение, графику какого качества мы хотим обеспечить в нашей игре.

Принципиальное отличие от геймплейных прототипов состоит в том, что даже на раннем этапе, когда еще может не быть проработана игровая логика, арт-прототип нужно делать на

выбранном движке. Если вы сумели продемонстрировать запрашиваемый уровень графики на CryEngine, нет никаких гарантий, что такую же картинку покажет Unreal Engine 4. Проблемы на данном этапе вполне могут быть аргументом для того, чтобы отказаться от одного движка в пользу другого.

В результате принятых решений необходимо создать арт-прототип – демонстрационную сцену. В зависимости от поставленных задач мы таким образом проверяем, могут ли наши художники красиво нарисовать нужную картинку, или, при другом подходе, оцениваем технические возможности нашего движка, сделав высокополигональную^[49], но при этом совершенно необязательно красивую, модель.

Такие **ТЕХНИЧЕСКИЕ ПРОТОТИПЫ** проверяют производительность или технические идеи. В первом варианте методы те же, что и с арт-прототипом, – собрать сложную, нагруженную сцену и посмотреть, как все работает на целевом железе. Это необходимо для понимания, насколько мы сможем оптимизировать игру к моменту запуска. Особенное внимание здесь уделяют сложным сущностям: большому количеству однотипных объектов в кадре, сложным анимационным системам, разрушаемым объектам.

Технические прототипы создаются также для отработки конкретной технической части игры, например сетевого протокола. Допустим, нам нужно проверить, как ведет себя игра при медленном интернете или на слабом компьютере. Для проверки многих технических составляющих существуют готовые решения, так что нет необходимости ехать к бабушке в деревню в поисках старого компьютера или плохого интернет-соединения.

По ходу любой разработки приходят новые идеи и их решения, так что запросы к технической части обычно растут. Здесь есть две крайности, из-за которых могут появиться проблемы. Делая все только по ТЗ и не учитывая возможное расширение функционала, можно столкнуться с ситуацией, когда любое отклонение повлечет за собой коллапс и необходимость переделывать все сначала. Лучше

заранее предусмотреть возможность расширения, например, максимального количества участников сессии, или слотов персонажа, или других множимых сущностей.

С другой стороны, стараясь сразу создавать максимальные возможности для любых задумок гейм-дизайнеров или для миллиарда онлайн-игроков, вы, скорее всего, потратите время и силы на нечто, что никогда не будет реализовано. Поэтому, чтобы придерживаться золотой середины, следует провести исследование, выслушать все запросы других специалистов и, проанализировав данные, умножить требования, к примеру, вдвое.

Технические прототипы также часто нужны, чтобы понять, сможем ли мы реализовать ту или иную физическую модель. Физика в играх – вещь для игроков сама собою разумеющаяся: мало кто задумывается о том, как непросто воссоздать реалистичное взаимодействие разных предметов. Скажем, мы хотим, чтобы в нашей игре была возможность честно топить предметы в воде. Необходимо проверить, можем ли мы написать физическую модель, при которой предметы различной массы и плотности станут по-разному тонуть.

Арт и технические прототипы часто объединяют, чтобы убедиться, что, с учетом имеющихся у нас ресурсов, можно создать красивую картинку необходимого качества. Здесь нет никакого геймплея: важно проверить, что самые тяжелые и сложные модели, виды освещения, разные ракурсы и т. д. смотрятся органично, воспроизводятся, как задумано, и ничего не ломают.

Прежде чем запускать полноценный продакшен, имеет смысл протестировать и **ПРОТОТИП ИНТЕРФЕЙСА**. Его также можно собрать на базе Figma, Flash или другой программы, помогающей предварительно оценить удобство и доступность интерфейса. Его можно собрать даже с помощью бумажных прототипов, хотя это более опасный путь, чем в случае с геймплейными прототипами, так как в этом вопросе большую роль играют именно ощущения. Лучше

проводить тесты с реальными размерами, разрешениями и артом, чтобы проверить, как это будет выглядеть в конечном продукте.

Прототипы интерфейсов имеют другие задачи. Прежде всего, тестируется возможность реализации всех задумок в рамках одного экрана, насколько удобно и красиво выглядят наши решения. Во-вторых, проверяем, правильно ли игрок понимает расставленные дизайнерами акценты, насколько просто и быстро он находит нужные ему функции. И, в-третьих, нужно убедиться, не вызывают ли наши решения ненужные паттерны поведения у игрока.

Например, главная часть интерфейса гиперказуальных ^[50] игр – это управление. Эти простые по своей сути продукты, сделанные в первую очередь для нон-геймеров ^[51], должны предоставить игроку максимально простое и понятное управление. Понятность заключается в использовании исключительно знакомых жестов: это прокручивание, как у ленты мобильных версий социальных сетей, например Instagram, это листание в стороны, клики и удержание касания. Непривычные пользователям жесты, типа двойного клика или многоходового использования кнопок, сильно усложняют продукт для казуальной аудитории.

Что же касается термина «простота в управлении» для гиперказуальных игр, то здесь имеется в виду однокнопочность управления и возможность играть в любом месте. Для этого мы используем вертикальную ориентацию экрана, привычную по мессенджерам. А также не требуем совершать составных действий. К примеру, считается хорошим тоном использовать в игре только одно из двух действий: прицеливание или выстрел. То есть либо игра прицеливается автоматически, а игрок решает, когда стрелять, либо игра стреляет постоянно автоматически, а пользователь лишь выбирает цели.

Будет вполне логично, если ваша команда станет работать над геймплейным, художественным и техническим прототипом одновременно.

КОМУ ПОКАЗЫВАТЬ ПРОТОТИП

Любой прототип, кроме совсем технических, должен помочь понять, какие ощущения вызывает игра, но во многом они субъективны. Без UX-тестирования довольно трудно отследить, в каком месте игроки могут столкнуться с затруднениями. И здесь очень важен выбор, на ком тестировать еще не готовый продукт.

Прежде всего, конечно, разработчики могут играть самостоятельно – для них отсутствие красивой графики и неработающая часть функционала не станут препятствием для оценки геймплея. Это необходимый этап, ведь если основной геймплей кажется неинтересным даже самим создателям, то очевидно, что игру нужно дорабатывать. Но даже если разработчики всем довольны, они обычно слишком хорошо знают свой продукт, чтобы объективно оценить, как он воспринимается на данном этапе.

Поэтому игру показывают потенциальным игрокам. Даже инди-команда может попросить довольно большое количество игроков ознакомиться с прототипом, но здесь все будет очень сильно зависеть от выбранной группы людей. Если это ваш хороший товарищ, еще и любящий подобные игры, ему будет крайне сложно оставаться непредвзятым. Поэтому старайтесь поощрять критику, просто попросив о честном отзыве или же пообещав бутылку любимого напитка тому, кто найдет больше всех недочетов.

Совершенно иначе обстоит дело, если мы хотим показать прототип широким массам. Чтобы отзывы были честными, зачастую не имеет смысла указывать, что это наша первая игра, над которой мы год трудимся в одиночку в гараже и на «Дошираке». Да, придется столкнуться с непониманием того, что это всего лишь прототип, половина функций пока недоступна и, вообще, «в GTA-5 графика круче». Но если обозначить все известные проблемы и попросить не обращать на них внимания, получить адекватную комплексную оценку будет невозможно. Так что придется стиснуть зубы, спокойно

принять все отзывы, предупредив, что сейчас вы демонстрируете именно прототип, не проверяете качество графики, а лишь, например, хотите узнать мнение о core-геймплее.

Прототипы не помогут оценить, насколько игра получится коммерчески успешной, потому что слишком многих составляющих еще не хватает и многое в результате исследований будет выброшено.

Прототипы очень редко показывают журналистам и стримерам. Первое впечатление очень трудно исправить, поэтому нужно четко осознавать, зачем показывается прототип. Но если игра построена на необычной идее и мы планируем привлечь за счет этого много органического трафика^[52], чтобы заранее заинтересовать издания или стримеров и начать создавать комьюнити уже на этапе прототипа, это может быть оправданно. Но, так как это еще не полноценный показ, обычно лучше договориться о нераспространении информации.

Показ прототипа инвестору – это сложный этап как для инди, так и для крупной компании. Делать игру для игроков или делать игру, которая понравится инвестору, – зачастую это решение имеет очень тонкую грань. И инвесторы, и разработчики – обычно нецелевая аудитория создаваемого игрового продукта. Если инвестор не сильно погружен в мир видеоигр, для него аргументы о результатах фокус-тестов на целевой аудитории, исследованиях рынка и конкурентов могут быть решающими, чтобы поверить в продукт. Большинство инвесторов понимают, что, не являясь, к примеру, 40-летней домохозяйкой, для которой создается игра, они вряд ли смогут адекватно оценить, интересен ли геймплей. Но, к сожалению, как и в любом бизнесе, бывает всякое.

Часто в игровых студиях человек, который презентует проект инвестору, и тот, кто представляет его аудитории, – разные люди, ведь для этого нужны разные навыки.

Итак, создавая прототип, мы выбираем средства, начиная с самых быстрых и дешевых. Для крупной компании вполне нормально на

протяжении нескольких месяцев проверять разные идеи и методы силами, например, пяти человек, ведь впоследствии эта работа может сэкономить время для пяти сотен сотрудников. В случае инди, работая над прототипом, обычно нет смысла долго разбираться с незнакомыми технологиями, которые не факт, что потребуются в дальнейшей работе над самой игрой.

Большинство прототипов не пригодятся вам в будущем. Следует относиться к ним как к получению опыта и знаний; это не конечный продукт, а лишь способ вовремя ответить на важные вопросы о проекте. Цель прототипа – не только проверить гипотезу, но и снять риски. Он помогает собрать все шишки, понять, как не надо делать и какие вещи вызывают наибольшие трудности, набраться опыта и наладить взаимодействие между членами команды, не затрачивая много времени и ресурсов.

Со временем вопросы, на которые должны ответить прототипы, будут становиться все более конкретными. Для отдельных механик подходят бумажные или написанные в простых программах прототипы; чем больше мы приближаемся к проверке полноценного геймплейного взаимодействия, тем сильнее нам нужны игровые движки; в случае же, когда мы проверяем сложную физику, уже не справиться без технического прототипа – часто с артом, созданным на ассетах, близких к реальным. Очень сложно с помощью готовых ассетов проверить, не будет ли, например, игрока укачивать, пока он будет уворачиваться от монстров в нашем VR-шутере. Здесь нельзя обойтись геймплейным прототипом, он не ответит на этот вопрос.

И главное – нельзя долго задерживаться на этапе прототипирования. Если он затянулся, значит, скорее всего, нужно изменить подход к созданию прототипа либо признать проверяемую идею неудачной.

Прототип указывает на те проблемы, о которых вы раньше не подозревали. Это тест вашей игры, вашей команды, и итог этого теста вполне может быть неудачным. Порой бывает сложно расстаться с плодами своих трудов, но, как и в любом исследовании,

даже отрицательный результат – тоже результат. Цель прототипирования – получение информации, а не готовая игра.

Если же гипотезы подтвердились, с помощью прототипов команда может приступить к созданию вертикального среза игры, включающего в себя образцовый уровень, на который вы будете ориентироваться в дальнейшем.

Теперь вы примерно представляете, сколько времени и ресурсов требуется вашим специалистам для создания, например, игровой фичи или анимации объекта. Суммировав полученную информацию, вы сможете оценить общую длительность разработки и предположить дату выхода вашей игры, а команда получила опыт, необходимый для того, чтобы спланировать следующую стадию – продакшен.

Продакшен

Процессы этапа продакшен

На этом этапе нам предстоит создавать уже не прототипы, а полноценную игру. Как правило, этот этап самый длинный, поэтому важно правильно настроить все процессы.

КЛЮЧЕВЫЕ ЦЕЛИ ЭТАПА ПРОДАКШЕН

Вот что нас ждет:

- производство контента;
- вертикальный срез (от англ. vertical slice) – это один или два образцовых уровня вашей игры, которые можно демонстрировать заказчику/игроку. По сути этот этап находится ровно между препродакшен и продакшен. С одной стороны, по его итогам еще возможны значительные изменения, а произведенный контент может не использоваться в готовом продукте. С другой стороны, vertical slice создается уже с полноценными процессами, в хорошем качестве, по его образцу создаются остальные уровни;
- документация. Куда же без нее? Гейм-дизайн-документ (ГДД), маркетинговый план (во время релиза этим заниматься будет уже некогда);
- повседневное управление проектом и решение возникающих проблем;
- корректировка планов и прогнозов, составленных на прошлом этапе;
- результатом этапа продакшен должна стать готовая к демонстрации пользователям версия игры (бета-версия).

Даже успешные проекты на этом этапе производства сталкивались с проблемами, которые отбрасывали их на этап препродакшен. Нужно быть морально готовым к такому повороту событий. Вся

разработка игры, от идеи и до релиза – это многократная проверка гипотез, и неудачи могут случиться на любом этапе.

ПРОЕКТНОЕ УПРАВЛЕНИЕ НА ЭТАПЕ ПРОДАКШЕН

Процессы, построенные на предыдущем этапе, – это основа. Работа с процессами этапа продакшен – не про их построение, а про борьбу с текущими проблемами.

Допустим, наш проект покинул художник, и мы срочно должны найти нового. Нам повезло, и замену удалось отыскать быстро, но он считает, что выбранный арт-стиль не соответствует целям, и предлагает свой. Команде новый арт нравится больше. Что делать? Заставить новичка работать по утвержденным референсам вроде бы неправильно – никому уже не нравится старый вариант, качество не выросло, плюс мы демотивируем нового сотрудника. Перерисовывать заново – равно сорвать все сроки. Сохранить наработки обоих художников – любому заметно, что над артом работали разные специалисты, выглядит некрасиво.

Любое решение в данном случае плохое. На практике, чтобы минимизировать его последствия, есть вариант задуматься о том, можем ли мы позволить себе еще одного художника в помощь, чтобы, не срывая сроки, переделать контент. Могут ли гейм-дизайнеры и сценаристы придумать что-то, чтобы сохранить в игре оба арт-стиля и чтобы это выглядело органично. А насколько дорогим окажется их предложение?

Здесь не существует правильных ответов, а подобные случаи встречаются постоянно. На первоначальном этапе мы составляем список рисков, чтобы заранее продумать, как будем решать те или иные проблемы, даже понимая, что все предусмотреть невозможно.

Для инди-разработчиков продакшен – это, наверное, самое тяжелое время. Многие команды гибнут именно на этом этапе. Людей обычно становится больше, сильно давят внешние факторы

(решения конкурентов, сроки), старые сотрудники покидают команду, процессы и менеджмент усложняются. Творчество и фантазия, вдохновлявшие на первом этапе, сменяются планомерной работой. До релиза еще долгий путь, а перед глазами только море рутинных задач, так что мотивация сотрудников – важнейшая часть этапа продакшен.

Работа по майлстоунам помогает с мотивацией. В идеале процессы должны быть построены таким образом, чтобы сначала сделать главные фишки, снять риски и чтобы ни один сотрудник при этом не простаивал. Так что каждый специалист всегда в курсе, над чем работает остальная команда.

Работа над построением процессов ведется непрерывно, особенно при использовании Scrum. Все упирается в вопрос, что ждать от людей: если просить мало, скорее всего, и сделано будет мало, а если требовать много, команда будет нервничать, что не выполняются планы. Гибкие инструменты хороши, но всегда есть риск свалиться в хаос. Так что процесс по корректировке пайплайнов [\[53\]](#) ведется постоянно.

АУТСОРС

На этапе продакшен разработчики часто привлекают сотрудников на аутсорсе. Если вы работаете с такими внештатными сотрудниками, особенно важным становится утверждение, что без ТЗ результат будет довольно сомнительным. Критически важно грамотно составлять задания, так как внешние сотрудники могут неправильно истолковать требования. Еще одна банальная истина: необходимо четко договориться об условиях, сроках, штрафах и ответственности исполнителей и зафиксировать эти договоренности в документах.

На аутсорс часто отдают работы над артом. Если есть понятный пайплайн, нет смысла временно нанимать в штат много художников.

Например, большая часть танков для *World of Tanks* создается именно аутсорсными сотрудниками. На Западе работу над музыкой и игровым сценарием также принято отдавать на аутсорс, чтобы сотрудники не простаивали. Если же удастся договориться с известным автором, его имя станет еще одним USP проекта, как было, например, с Крисом Авеллоном для *Pathfinder: Kingmaker*. Локализовать (перевести на разные языки) игру своими силами – тоже неоправданно сложное занятие, лучше обратиться к специализированным внешним компаниям. Если игровой проект предполагает изолированные уровни, даже работу над левел-дизайном можно переложить на внештатных исполнителей.

Звук тоже часто отдают на аутсорс. Саунд-дизайнер – это человек, который занимается созданием всех звуков в игре. Музыкой обычно занимается отдельный специалист – композитор, но в России и странах СНГ нередко один человек отвечает за оба направления.

Саунд-дизайн можно разделить на две части: непосредственно создание звуков и техническая часть (встраивание звуков в движок игры с помощью сторонних программ или самого движка). По большому счету саунд-дизайн в играх и кино ничем не отличается, даже по этапам производства. Сначала записывают исходный материал, редактируют его под визуальный ряд или техническое задание и экспортируют в проект.

Для небольших проектов, не предполагающих большого количества звуков и их обработки, можно записать звуки самостоятельно или поискать что-то подходящее на многочисленных аудиостоках, в том числе бесплатных (например, Freesound или Sonnis). Для более крупных проектов обычно приглашается саунд-дизайнер, обладающий знаниями, как из разрозненных звуков создать атмосферную игровую вселенную. Работая с саунд-дизайнерами, очень важно четко доносить мысль, что именно в звуке вам не нравится; даже простая имитация желаемого звучания с помощью рта порой может быть информативнее, чем три страницы

текста. Очень часто проще самому один раз мяукнуть, чем пытаться объяснить на бумаге, какие звуки должна издавать кошка в игре.

Что касается музыки, ее тоже можно записывать самостоятельно, обладая необходимыми навыками и оборудованием, но чаще всего и эта работа отдается на аутсорс, так как это помогает сильно сэкономить. Если разработчики берут композитора или саунд-дизайнера в штат, его необходимо обеспечить отдельным тихим помещением, где он сможет спокойно трудиться, музыкальными инструментами и специализированным софтом. Нужно иметь в виду, что, например, одна библиотека оркестровых сэмплов может стоить от 3000 евро. Если же вы работаете с аутсорсным композитором или саунд-дизайнером, все необходимые программы и оборудование у него уже есть.

А вот если отдать внешним исполнителям работу над ключевыми техническими решениями, базовой инфраструктурой или гейм-дизайном, это может привести к неприятным ситуациям, когда все штатные сотрудники простаивают из-за неисполнительности сотрудников на аутсорсе. Если определенные специалисты нужны на всех стадиях работы, им требуется большая степень погруженности в проект и беспрепятственная коммуникация с командой, лучше держать их в штате.

Внутри офиса обязательно должны быть специалисты с проверенной экспертизой в сфере работы с внешними исполнителями (арт, звук и прочее). Например, без хорошего арт-директора в штате очень сложно оценить, выполнены ли задачи в полном объеме и что необходимо откорректировать. Существует даже отдельный вид аутсорса – однократный наем опытного консультанта, который приглашается на проект, чтобы сказать, что мы делаем не так. Допустим, у нас в игре планируется сложная физика, и штатный технический директор, разбирающийся в вопросе, будет стоить очень дорого. Иногда оправданно набрать в команду сотрудников попроще, и, когда мы набьем первые шишки, один раз оплатить работу опытного специалиста, чтобы он

подсказал, что нужно исправить. Такой подход часто оправдан для инди-команд, испытывающих проблемы с недостатком экспертизы, – особенно учитывая, что такую консультацию можно получить и бесплатно, участвуя в различных конференциях и программах.

Другая частая ошибка работы с аутсорсом – непрозрачность процесса работы, отсутствие контроля над исполнителями. Это создает ряд проблем: либо мы можем получить не то, на что рассчитывали, либо, если мы не разбираемся в теме, переплатить. Работать с несколькими исполнителями – хорошая практика, так как мы можем сравнивать результаты и стоимость, чтобы сделать оптимальный выбор. Иногда студии предоставляют услуги даже дешевле, чем отдельные фрилансеры^[54]. Качество у них, как правило, выше, ведь все процессы налажены и есть некая взаимозаменяемость (если сотрудник заболел, его подменит коллега, и заказ все равно будет выполнен). Еще вариант: выложить задание на специальных ресурсах и договориться с человеком, предложившим оптимальные условия.

Игровые механики и правила игры

Взаимодействие сложных систем, составляющих видеоигры, заставляет причастных поднимать комплексные вопросы: глобальные цели гейм-дизайна, определяющие механизмы игр и ожидания от игрового процесса. Дизайн, программирование, нарратив, психология игрока сливаются в единую и нередко непредсказуемую цепь игровых событий.

Игровые механики – не просто свод правил. Алгоритмы сами по себе, в отрыве от ожидаемых эмоций, не обеспечивают вовлечения. Вы должны позаботиться о том, чтобы игрок всегда мог ответить себе на три вопроса: «Что я должен делать в игре прямо сейчас?», «Как долго я буду в это играть и вернусь ли в игру, например, завтра?» и «Зачем я вообще играю в эту игру?» Это поможет гейм-дизайнеру обозначить краткосрочные, среднесрочные и долгосрочные цели.

Возможность выбора решений, от которых зависит исход игры, – одна из ключевых целей гейм-дизайна. Когда человек проигрывает и не понимает, где он ошибся, значит, игровой процесс не до конца ясен и это ошибка гейм-дизайнера. Информация должна быть достаточной и своевременной, доступ к ней должен быть легким, прочтение – однозначным. У игрока должна быть возможность просчитать вероятность того или иного результата своих действий.

Другая крайность, когда игроки видят, что игра «помогает» им победить, это вызывает не меньшее разочарование и агрессию. Игрок должен видеть причинно-следственную связь между своими действиями и результатом (не обязательно положительным). Значимость решения – вот что делает игру привлекательной и интересной.

Игровой мир должен давать четкую реакцию на действия пользователя. Именно это правило позволяет вовлечь человека настолько, чтобы он поверил в реальность происходящего на экране

и забыл о существовании этого самого экрана. Не так важно, будет ли результат такого действия обозначен всплывающим окном или же, свернув на развилке направо, игрок столкнется с бандой головорезов.

Лучше предусмотреть, чтобы игрок понимал последствия своего выбора. Например, в легендарных *«Героях Меча и Магии III»*, в отличие от последних частей, игрок без специального умения не видит силу отряда противников, на который может напасть. Если игрок по каким-то причинам еще не досконально изучил бестиарий игры, откуда ему знать, ждет ли его легкая победа или неминуемая гибель?

Чтобы игрок чувствовал себя вовлеченным и ответственным за решения, игра не должна подкидывать такие неприятные сюрпризы. В русской локализации *«Ведьмака 3»* есть эпизод, когда нужно сделать очередной выбор, как ответить Дийкстре. Вариант «сильно оттолкнуть Дийкстру» кажется вполне безобидным, а в итоге Геральт ломает ему ногу, после чего Дийкстра почему-то огорчается и не хочет давать дополнительные квесты. Когда игрок не может предсказать последствия своего решения, игра превращается в угадывание, «чего от меня хотел гейм-дизайнер», а это не способствует ни вовлечению, ни адекватности развития игровых событий.

Не отдельные средства выразительности (диалоги, визуальный стиль, музыка, правила игр и пр.) определяют опыт игрока. Сама реакция игры, интерактивность становится средством выразительности.

Для игрока – это опыт и эмоции, для гейм-дизайнера – средство их достижения; так что, работая над игрой, нужно помнить обе, часто несовпадающие, точки зрения. Пользователь может считать игру интересной или скучной, даже понятия не имея, что игровые механики вообще существуют, что они оказывают влияние на напряжение игрового повествования. Это одна из главных трудностей дизайна игр: до анализа результатов плейтестов нет

никаких гарантий, что все труды по созданию правил, воплощению деталей и идей вызовут у игроков именно то впечатление, на которое рассчитывал гейм-дизайнер. А так как он работает именно для игроков, придумывать правила и ограничения бывает не так интересно и весело, как в итоге играть в готовый продукт.

ДОБАВЛЕНИЕ НОВЫХ ИГРОВЫХ МЕХАНИК

Почти всегда новые механики делают геймплей более комплексным, сложным. Механики могут развивать существующие правила, усиливая значение тех или иных взаимодействий. Например, получив больше бонусов на первом уровне, вы легче пройдете второй; эффективное распределение ресурсов в стратегии на первых этапах игры дает накопительный эффект и преимущества. Результат: проигрывающему игроку очень сложно догнать лидеров, поэтому игра становится более динамичной и быстрее заканчивается. Такое положение вещей может расстраивать группу игроков.

Когда отрыв непреодолим, такая обреченность на поражение или победу делает продолжение игры неинтересным и бессмысленным. Поэтому дизайнеры могут вводить механики, нарочно замедляющие игровое развитие лидера или помогающие отстающим. Например, при определенном количестве жизни шанс на критический урон многократно возрастает, что помогает догнать соперника. Или же в *Mario Kart* отстающим доступны самые мощные подбираемые бонусы, помогающие временно выбить из строя лидеров и догнать их. Такая возможность закрыть глаза на первоначальные ошибки позволяет сократить разрыв, и финальный этап игры или части игры становится более значимым.

И те, и другие механики кардинально влияют на ощущения от процесса. И снова, только сыграв, можно понять, пошли ли внесенные изменения игровых взаимодействий на пользу,

способствуют ли они вовлечению, делают ли конкретную игру более глубокой и интересной. Когда в играх появились стрелочки, указывающие на карте путь до квестов, многие разработчики решили добавить эту механику в свои игры. Для больших игр, где много гринда^[55], это отличное решение; но для ряда проектов, где важную роль играли нарратив и исследование мира, такое нововведение просто убивало для части аудитории все удовольствие, ведь общение с персонажами и изучение локаций потеряли смысл.

В *World of Warcraft* раньше, чтобы собрать группу для рейда, нужны были чернокнижники, обладающие возможностью призыва, маги, создающие порталы, и пр. Сейчас собрать группу и телепортироваться в подземелье можно с помощью одной кнопки. Более казуальным игрокам стало проще, а вот другие тоскуют по особой социальной составляющей игры, связанной со сложной механикой сбора групп. Но в результате эта фишка привлекла в подземелья много казуальных игроков, так что свою задачу она выполнила. Поэтому ввод любых изменений игровых механик должен иметь четкие цели и тщательно тестироваться.

Если гейм-дизайнер добавляет или меняет какую-то механику, не понимая, какую пользу это нововведение принесет игроку, он выполняет очень странную и зачастую бесполезную работу. Например, зачем вводят криты (критический урон)? Ведь можно просто планомерно, с каждым ударом, усиливать урон, и математически результат будет одинаковым. Ответ: происходящие игровые события становятся менее однообразными, у игрока появляется яркий момент, а математически просчитать влияние этой механики так же просто.

Вы хотите ввести в своей игре возможность скрытного прохождения. Зачем? Разнообразить геймплей? Да, но это общие слова. А, например, дать возможность спрятаться от врагов и передохнуть, не выходя из приложения, – действительно аргумент в пользу введения этой механики. Часть игроков предпочитает именно

такой геймплей, и стелс станет стимулом играть дальше. Или же появляется возможность перепрохождения за счет уникальных игровых ситуаций, доступных только в этом режиме, – вы должны четко обозначить, как изменения сделают игру лучше.

Левел-дизайн

Левел-дизайн включает в себя создание уровней для игр – локаций, окружения. Его можно определить как специфический прикладной навык на стыке гейм-дизайна и арта. Задача левел-дизайнера (обычно на этой должности находится отдельный специалист) – создать игровой мир таким образом, чтобы, во-первых, реализовать в его рамках все задуманные типы геймплея, а во-вторых, передать правильное ощущение от картинки. Можно выделить три подхода к левел-дизайну, последний из которых самый сложный.

Первый вариант – мы просто собираем геймплей в игровые уровни. Примером могут служить некоторые головоломки и матч-3, где как такового игрового мира нет, есть лишь набор механик и сеттинг. Задача – донести до игрока цель игрового уровня и способы ее достижения.

Второй вариант – когда нам важно с помощью правильно собранного уровня передать игроку нужные ощущения через арт. Такое часто бывает в квестах: герой бродит по локации, находит предметы, применяет их, переходит в следующую зону. Для файтингов окружение – тоже обычно декорация. Задача левел-дизайна здесь – верно расставить красивые и атмосферные игровые объекты, чтобы игроку было с чем взаимодействовать. Во многом это работа именно по интеграции арта в игру, ею может заниматься художник.

И третий, продвинутый вариант – совмещение двух предыдущих подходов. Создать не просто декорации для действий игрока, а живой игровой мир. Цели здесь более комплексные: объяснить правила игрового мира, органично реализовать в нем игровые механики, подтолкнуть игрока к правильным решениям, не забывая о динамике и ключевых ощущениях. Левел-дизайн сильно привязан к конкретному игровому жанру и создается по разным принципам.

Возьмем, к примеру, **MMORPG**. Допустим, необходимо создать новую локацию – тропический остров. Основную идею и ощущение от этой локации определили как «мир – не то, чем он кажется». Поэтому, прибыв на остров, игроки видят красивые пейзажи, райские пляжи, белый песок, встречают дружелюбных аборигенов, но замечают некоторые странности, намекающие на тайны этого места. Немного свернув с тропы, игрок видит несоответствия первому впечатлению об острове, а квесты дают возможность разобраться, что здесь на самом деле происходит. Исследуя локацию, персонажи должны прийти к выводу, что на самом деле они попали в виртуальную реальность, куда их поместили, чтобы высасывать их энергию, и нужно срочно отсюда выбираться.

Когда концепция определена, левел-дизайнеры приступают к подготовке первых набросков. Они рисуют карту острова, где обозначено, по какому маршруту, проходя квесты, игрок будет перемещаться: вот в этой точке он узнает, что попал в виртуальный мир, здесь – попадет в реальность и поймет, что на самом деле люди живут в капсулах, вот сюда он вернется, чтобы спасти товарищей, и т. д. Это первый скетч, просто картинка, где можно увидеть предполагаемый путь по локации.

На следующем этапе эта картинка должна превратиться уже в игровой уровень, но пока без финальной графики. Это геймплейная болванка, окружение собрано из коробочек разных цветов, вместо персонажей – манекены. Но с ними уже можно взаимодействовать, убивать этих картонных монстров, гулять по локации – так появляется понимание, что должно окружать игрока в тот или иной период времени. Параллельно с созданием такого прототипа левел-дизайнеры вместе с художниками определяют, как в итоге должно выглядеть это место.

Далее в эту болванку поэтапно добавляют уже финальный арт: расставляют персонажей, здания, предметы. На этом этапе левел-дизайнеры начинают заказывать у художников некие объекты, цель которых – передать нужные ощущения. Сначала структурную

геометрию, например красивый храм, в который можно войти и что-то обнаружить. Теперь эту карту можно тестировать и показывать игрокам, чтобы настроить баланс: избежать моментов, где игроку скучно из-за отсутствия монстров либо из-за утомительной дороги до квеста, либо эта дорога вообще проложена не оптимально. После этого можно добавлять неважные для геймплея пропсы^[56] (морские звезды, красивые пальмы, необычные статуи и прочее).

MMORPG обычно не предполагают долгую отладку локации, задача левел-дизайна здесь – просто развлечь игрока, пока он выполняет задания. Однако следует помнить об особенности жанра: в одной локации может находиться сразу множество игроков, поэтому важно избегать «узких» мест, способных затормозить прохождение уровня.

ШУТЕРЫ И СЕССИОННЫЕ ИГРЫ, предполагающие победу или поражение, имеют другие акценты. Балансировка карт здесь обычно занимает куда больше времени, так как крайне важно, чтобы условия были честными, а окружение давало возможность использовать разные игровые механики. Первый этап аналогичный: выбрать референс или идею для карты: война в Ираке (пустыня и нефтяные вышки), Сибирь (снег, типичная российская деревушка) или пиратский остров.

Этап геймплейной болванки в случае шутеров и сессионных игр должен показать, какие типы геймплея будут реализованы в рамках игровой сессии, исходя из заданного окружения.

Никто не будет играть в шутер на ровной поверхности – важно, чтобы были укрытия, препятствия, позиции для засады. В одной части карты можно найти выгодные позиции для снайперов, а вот здесь, если быстро пробежать, можно, например, обойти вражескую команду и настичь ничего не подозревающих противников на полянке.

Допустим, у нас сессионная игра про танковые сражения. Результаты работы над левел-дизайном могут быть такими: если игроки поедут направо, геймплей будет про то, чтобы прятаться в

лесах и стрелять оттуда: налево – город, узкие улочки, где можно организовать засаду и обходить противников сзади; прямо – игрока ждет лобовой бой на мосту, здесь нет укрытий, и его ждет яростная перестрелка.

Финализация прототипа карты для этих жанров занимает много времени. Можно три месяца создавать локацию и еще полгода балансировать ее. Шутеры очень чувствительны к правильному окружению, ведь можно убить кого-то с дистанции за одну секунду, и, если у нас неправильно расставлены дома или другие виды укрытий, позиция сразу имеет ощутимое преимущество. Из десятков карт (в таких играх, как *Counter-Strike*) популярны лишь несколько, так как задача по балансировке карты для сессионных игр – серьезный вызов. Попробуйте поиграть в *Counter-Strike*, замеряя время, требуемое, чтобы занять выигрышные позиции для обеих команд. Это в разы улучшает понимание, какое расположение является для команд выгодным, и игровой таймер приобретает новый смысл. Можно точно рассчитать, через какое время соперник окажется в той или иной точке, и, например, кинуть туда гранату. Левел-дизайн и расчет тайминга в этой игре можно считать образцовыми.

Прежде всего необходимо, чтобы, даже в сотый раз играя на одной и той же карте, вы могли использовать разные стратегии для победы. Во-вторых, карта должна читаться: даже новичку должно быть интуитивно понятно, что для того, чтобы его не убили, нужно спрятаться за эти ящики. Для таких игр геймплей в приоритете, и объекты создаются именно для него, даже если они неоправданны с точки зрения композиции, географии или сюжета. И несмотря на это, важно стремиться к ощущению естественности происходящего.

В **ОДИНОЧНЫХ ПРИКЛЮЧЕНИЯХ** (в духе *Uncharted*) все работает по-другому. Здесь левел-дизайн и определяет весь гейм-плей, и передает палитру ощущений от игрового мира. Ключевой элемент здесь – кривая сложности, когда моменты напряжения компенсируются расслаблением и наоборот, что делает игру

интересной. Допустим, вы пришли в новую локацию, набежали монстры, от которых едва получилось отбиться; теперь можно спокойно осмотреться, порешать головоломки, пособирать ресурсы, насладиться красивыми видами, перевести дыхание.

Большое внимание уделяют ощущениям игрока: что он может ожидать, а что нет. Если, убивая противников, мы получаем с них много хорошего снаряжения, можно сделать вывод, что мы двигаемся к боссу и игра помогает подготовиться к сложному сражению. Для хорроров левел-дизайн может быть даже важнее геймплея: внезапное появление страшных тварей в момент, когда игрок этого не ожидает, и недружелюбное темное извилистое пространство – основа таких игр.

Хороший левел-дизайн стимулирует игрока исследовать мир. Если мы просто поместим сундук с сокровищами в лес – никто не будет его искать. Даже каким-то чудом узнав о его существовании, прочесывать весь лес квадратами – долго и скучно. А вот если, просто гуляя по лесной тропинке, мы видим какие-то поваленные деревья, это может вызвать любопытство и желание узнать, что там скрывается. Оказывается, это бандиты организовали засаду (срубили деревья, чтобы не проехала телега); убив их, игрок находит записку о сундуке с сокровищами, спрятанном в лесу у водопада. Водопада он до этого не встречал, но можно предположить, что он должен быть где-то на возвышенности неподалеку, и как раз с этого места виднеется какой-то холм. Поднимаясь на него, мы слышим шум воды, и можно сделать вывод, что идем мы правильно; у водопада нас ждет босс-файт с главарем банды и его друзьями, победив которых, мы находим заслуженную награду.

Это мини-квест: никто не говорил игроку, что делать, не было заданий в журнале или знаков вопроса на карте; просто естественное событие в лесу, потому что игрок обратил внимание на странные деревья.

Однопользовательские игры – это часто набор завязок. Игры девяностых годов были достаточно сложными и нишевыми;

в нулевых, когда играми стало интересоваться больше людей, сложность существенно снизилась. Игрока не просто вели за руку, если он сворачивал с дороги или делал что-то не так, его буквально насильно затаскивали обратно, нередко с помощью левел-дизайна. Можно раскладывать интересные объекты таким образом, чтобы, собирая их, игрок обязательно пришел в нужное дизайнерам место; кровавые следы ведут к трупам – завязка детективной истории; указания могут давать NPC^[57] или всплывающие окна – вариантов довольно много. Появление монстров – тоже прекрасный инструмент, чтобы замотивировать игрока либо подойти и их убить, либо убежать. Поэтому обычно врагов расставляют в значимых точках, но таким образом, чтобы игрок мог спокойно перемещаться, не натываясь на них на каждом шагу.

Сегодня хорошие примеры, как можно направить игрока, – это *The Legend of Zelda: Breath of the Wild* и *God of War*. В этих играх нас подталкивают к базовому контенту, но без насилия: точки интереса разложены по локации, есть и сложные задачки, и открытый мир, и боковые ответвления для любителей исследовать игровые миры.

Противопоставляют два подхода: есть полный открытый мир в духе *Minecraft*, где можно делать и менять все, что захочешь, а развлечения игрок придумывает в этой песочнице себе сам; и есть жесткий коридор, с которого нельзя свернуть, зачастую нет возможности даже пойти назад, зато здесь все просчитано – монстры набегает только в нужном количестве, а награда соответствует усилиям. В современных больших играх пытаются это совмещать: есть прямой путь для казуальных игроков и большое количество интересного контента для любителей ответвлений от основного сюжета. Если такие сайд-квесты не нужны для выполнения глобальной цели данной локации, стандартом левел-дизайна стало предусмотреть какой-то дополнительный бонус за приложенные усилия.

ПРОЦЕДУРНАЯ ГЕНЕРАЦИЯ УРОВНЕЙ

Отдельная тема – это процедурная генерация уровней. Здесь гейм-дизайнер не создает уровни, а задает правила, по которым они должны сгенерироваться. Процедурная генерация может быть нескольких уровней.

1. Ее нет, и все сделано гейм-дизайнером вручную или с помощью игрового движка. В этом случае, открывая уровень в редакторе уровня, дизайнер будет видеть ровно то же, что и игрок.

2. Процедурная генерация есть, но на уровне разработчика. Левел-дизайнер может воспользоваться готовым решением по генерации уровня. Просто определить, что здесь должна быть пещера, а здесь – лес, локации создаются сами. Потом можно вручную подкорректировать что-то, расставив объекты; создать опушку, удалив деревья, или добавить труп какого-нибудь бедолаги, чтобы намекнуть на игровое событие. Чаще всего такой подход используется в матч-3 играх. Левел-дизайнеры рисуют поле, расставляют игровые элементы (механики, цели) и задают параметры для генерации фишек на поле, шансов их выпадения и т. д., а затем уже дорабатывают и проводят аналитику каждого уровня, чтобы проверить его играбельность и сложность прохождения.

3. Процедурная генерация может сочетаться с контентом, созданным вручную. Такие игры – еще не *No Man's Sky*, в них есть базовый геймплей со стандартными локациями, но можно найти и процедурно созданные элементы. Например, локация может быть фиксированной, но 30 % монстров, деревьев, домов и т. д. генерируются по-разному для конкретной игровой сессии. Что-то остается неизменным, и можно учиться на своих прошлых ошибках, а что-то оказывается в необычном месте, и перепрохождение получается более интересным.

Иногда разработчики оставляют возможность пройти локацию несколькими способами. В этом случае левел-дизайнер собирает все

эти 10 фиксированных вариантов уровня, а игрок случайным образом попадает на один из них.

Такой подход часто применяется как в классических RPG, так и в любых играх с наличием зон спавна^[58]. Это зоны на карте, которые состоят из некоторого количества точек спавна или являются их множеством. При старте игры, к примеру в некоторых популярных battle royale, алгоритм размещает разный лут^[59] в разных местах. Еще один популярный кейс – это *«Герои Меча и Магии»*, где существуют точки спавна артефактов, ресурсов и монстров. Однако что именно там появится, задается не обязательно жестко, а неким диапозонным вариантом. Это, безусловно, добавляет реиграбельности^[60] этой классической игре. Сюда же с некоторой натяжкой можно отнести все MOBA или action-игры, где каждая игровая сессия проходит на случайной карте.

4. И последний уровень, когда фиксированный контент используется по минимуму или не используется вовсе. Это не уровень, а просто алгоритм, по которому он создается. Такая генерация позволяет получить уникальный игровой опыт. Базовые движки не дают полностью сгенерировать рандомный уровень, нужны будут специфические решения.

Яркий пример – подземелья в классической серии Diablo. Каждый новый заход на уровень может отличаться от предыдущего, потому что его карта будет генерироваться случайным образом, собираясь из заранее заготовленных шаблонов. Тот же способ отлично используют гиперказуальные игры, предлагающие игроку бесконечную прогрессию. Алгоритм имеет заранее заданные параметры сложности и набор ассетов, что позволяет ему выстраивать новые и новые уровни.

Обычно, работая над последними двумя вариантами, левел-дизайнеры сначала создают типовой и предельный уровни, причем собирают их вручную. Например, при одной конфигурации уровень получится крайне сложным, при другой – почти элементарным; этот уровень – только про сражение с монстрами, этот – только про

головоломки и т. д. После этого программисты настраивают генерацию так, чтобы можно было взять любую точку в этих пределах.

При хорошем левел-дизайне чем больше контента сделано вручную, тем лучше он получается для первого прохождения, но реиграбельность его довольно низкая. А с процедурной генерацией – наоборот, такую игру интересно перепроходить снова и снова, но при разовом прохождении она будет уступать играм, созданным вручную, уделяющим большое внимание развлечению игрока.

БАЗОВЫЕ ПРАВИЛА ЛЕВЕЛ-ДИЗАЙНА

В отличие от гейм-дизайнера, левел-дизайнер должен много внимания уделять **ГЕОМЕТРИИ** уровня. Это физическое пространство уровня, на котором происходит геймплей, без арта, монстров и пр.

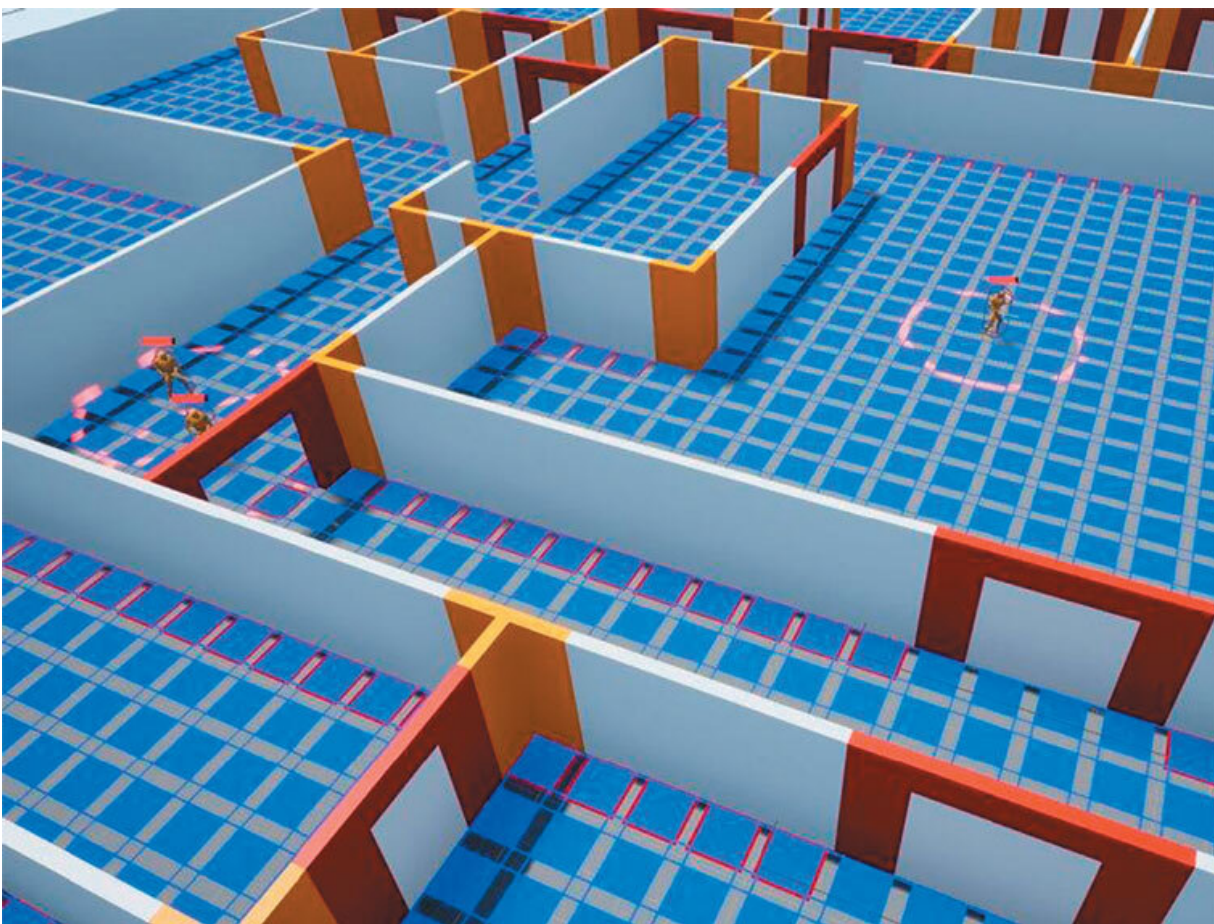


Рис. 15. Визуальный пример геометрии уровня

Именно геометрия вынуждает применять выбранные игровые механики, чтобы решить задачу уровня. Если игра предполагает несколько типов геймплея, но уровень построен таким образом, что выбирать один из них – неэффективно, значит, левел-дизайн недоработан. Допустим, персонаж может кидать гранаты, а локация представляет собой узкие коридоры с низкими потолками. Если в таких условиях кинуть гранату на дальнее расстояние нельзя, это оружие становится бесполезным, и механика не работает. Сталкиваясь с разными вызовами, игрок вырабатывает определенные шаблоны поведения. Так что нужно придумать очень хорошее объяснение, почему механика, которая много раз использовалась в аналогичной ситуации, перестала работать;

почему, например, точно умея прыгать, главный герой не может преодолеть заборчик высотой ему по колено.

Другая типичная ошибка: одна и та же механика повторяется на каждом уровне. Естественно, что игровые ситуации могут повторяться, но, чтобы вызвать у игрока новые впечатления, лучше добавлять дополнительный компонент. Допустим, в игре есть стандартный бой с противником; если менять только визуальную составляющую, это может наскучить, лучше добавить дополнительную механику. Например, тот же бой, но нужно использовать еще и прыжок, а затем тот же бой, но с двумя противниками сразу, бой с двумя противниками, но на движущейся платформе, и так далее.

Именно геометрия помогает построить правильную динамику игры, так как сама по себе задает игровые ситуации и делает интересным даже базовое передвижение по уровню. Нужно внимательно следить, чтобы у игрока оставалось достаточно способов преодолеть то или иное расстояние: ходьба, прыжки, кувырки. При этом возникает большая проблема, если игроку кажется, что проход есть, но на самом деле область недоступна.

Игрок не должен теряться на уровне. Если он долго не может понять, в какую сторону двигаться и что ему делать, скорее всего, он просто закроет игру. Лучше добавить запоминающиеся ориентиры, чтобы пользователь всегда знал, откуда он пришел. Неподсвеченные двери, которые невозможно обнаружить, или, наоборот, нагромождение игровых объектов, юнитов, декора сбивают с толку и могут сломать ощущение реальности происходящего.

РАБОТА С РЕФЕРЕНСАМИ – важная часть левел-дизайна. В первую очередь рассматривают примеры из реального мира. Вспомните соборы в *Assassin's Creed*; перенос реально существующих локаций и объектов в игровой мир может стать одной из важных особенностей проекта.

Можно брать за основу и левел-дизайн других игр. Собрав уровень с артом, вы получите первый вариант своей игры, с которым

можно будет дальше работать. Для больших проектов левел-дизайн часто собирают из отдельных кусков: пример деревни, пример бункера, пример полянки с монстрами и т. д. При попытке сразу создать огромную зону для открытого мира есть риск месяцами работать над чем-то, что может не пригодиться.

Бывает, что левел-дизайнер решает, как собрать уровень, исходя из набора утверждений от гейм-дизайнера. Важно заранее согласовать возможности персонажа: на какую высоту он может подпрыгнуть или залезть, какие способности у него должны быть открыты, чтобы преодолеть то или иное препятствие. Даже такая тривиальная вещь, как ширина дверного проема, может влиять на геймплей. Если в игре есть узкий проход и вы не можете убежать от монстра через такую дверь, это, скорее всего, станет проблемой. Размеры дверных проемов и коридоров будут зависеть и от расположения камеры. При виде от первого лица можно делать относительно небольшие проемы и узкие коридоры, но чем дальше камера от персонажа (вид от третьего лица, изометрия), тем больше должны быть габариты объектов.

Стоит подумать и о времени. Если мы договорились, что карту можно пробежать от края до края за 5 минут, важно определиться и со скоростью персонажа. Это может сильно влиять на геймплей: разница скорости в 1 метр в секунду может определять, нужно ли, например, тщательно прицеливаться или важнее быстро перемещаться в нужные точки.

Гейм-дизайнер задает правила игрового мира. Допустим, 50 % времени игрок должен стрелять, 30 % – перемещаться между точками, 20 % – лечиться, захватывать точки и крафтить оружие. Левел-дизайнер же должен превратить эти данные в игровой мир.

Если есть задача сделать его атмосферным и живым, не получится просто добавить красивый храм посреди поля. К нему должна быть проложена дорога; монахи что-то едят, значит, можно найти огороды, пастбища, сараи, лавочки и прочие элементы инфраструктуры, позволяющие поверить, что перед нами

действительно жилая постройка. Игры, претендующие на фотореалистичную графику, нередко повторяют реальные улицы с Google Maps, чтобы не забыть важные элементы. Такой тщательный подход, когда даже на случайном столбе можно прочесть газетную вырезку, намекающую на какое-то игровое событие, и отличает AAA-игры. Игры с пиксельной или стилизованной графикой, напротив, обычно лишь намекают на детали, позволяя игрокам додумывать их самостоятельно.

В итоге мир воспринимается живым, когда он либо тщательно проработан и детализирован, либо оставляет место воображению, то есть является более абстрактным, стилизованным. В противном случае мы воспринимаем игровое окружение просто как декорации, что тоже может быть оправданно, если игра не ставит задачу глубокого погружения в предлагаемую реальность.

Референсы, понимание потребностей аудитории, знание геймплейных особенностей и хорошие отношения с художниками позволяют левел-дизайнеру работать эффективно, создавая увлекательные и запоминающиеся игровые миры.

Документация: гейм-дизайн-документ

Итак, вы сумели создать vertical slice, поработали над отдельными фидами, и вас ждет этап полноценного производства. Теперь необходимо, чтобы по каждой фиде мы имели полное описание от гейм-дизайнеров и грамотные оценки от исполнителей и продюсера. На этом этапе вам предстоит составить гейм-дизайн-документ.

Не стоит стремиться запихнуть в гейм-дизайн-документ всю информацию о проекте. Обычно ГДД – это набор разных небольших документов, каждый из которых описывает определенную фиду или механику. Например, ГДД на механику перемещения.

Помимо того что ГДД помогает зафиксировать свои и чужие задачи по проекту, это еще один инструмент для записывания идеи таким образом, чтобы минимизировать вероятность ошибок. Именно по этой причине ГДД нужен на любом проекте, хотя многим такая бюрократия и может казаться излишней. Даже работая над игрой в одиночку, важно сформулировать ответы на ряд вопросов, чтобы не пропустить неверного гейм-дизайнерского решения.

Подходы к составлению документации для разных сотрудников, менеджеров и технических специалистов могут отличаться. Но суть везде одна: документ должен пояснять, какими средствами мы будем пользоваться для достижения тех или иных ощущений у игроков. Неважно, работаете ли вы над расчетом баланса или системой процедурной генерации уровня, гейм-дизайнер всегда должен держать в голове, зачем игроку то, что он описывает.

ГДД – это развернутое описание, как вы ведете пользователя к нужным ощущениям, для отдельной системы или для проекта целиком. На основании этого документа формируется (как часть ГДД) практическая инструкция для исполнителей – техническое задание (сокращенно ТЗ). Здесь хранится вся информация для программистов, художников, тестировщиков и т. д.

Обычно работа над фичей начинается именно с изменений в шаблоне ГДД. Такой шаблон встречается практически в каждой компании – гейм-дизайнер редко придумывает его с нуля, в этом нет смысла. Либо вы ведете документацию каждой версии, либо создаете отдельный документ для каждой фичи. Шаблоны для разных задач могут отличаться: например, шаблон для дизайна квеста, шаблон для дизайна абилки и т. д.

В шапке ГДД на отдельную фичу всегда обозначен человек, ответственный за нее, – **FEATURE OWNER**, – чтобы при возникновении вопросов любой член команды знал, к кому можно обратиться за пояснениями. Если фича уже реализована и по ней проведена аналитика, ссылку на ее результаты также прикрепляют в начале документа. Это конкретные, заранее продуманные метрики любых взаимодействий с фичей и их последствия. В некоторых компаниях заранее, еще на этапе составления ГДД, записываются метрики, которые должны отслеживаться; а когда готов аналитический отчет, он закрепляется в шапке или рядом.

Далее ГДД содержит **ИНФОРМАЦИЮ ОБ ИТЕРАЦИЯХ**, чтобы можно было сразу понять, на каком этапе находится работа над фичей. Например, итерация 0 – базовое описание функционала, итерация 1 – что нужно поменять/добавить и т. д. Для разных итераций могут заводить отдельные ГДД.

Итак, первым делом необходимо обозначить **КОНКРЕТНЫЕ ЦЕЛИ**, почему вы хотите ввести ту или иную фичу. Цели должны быть достижимыми и измеримыми, чтобы впоследствии проверить, достигли мы их или нет. Это нужно бизнесу и менеджменту, чтобы иметь информацию для принятия решения, делать их или нет, исполнителям, желающим лучше понять, для чего они работают над той или иной задачей, и вам, чтобы восстановить ход размышлений спустя какое-то время. Четкие цели помогают экономить время и не читать все ГДД в поисках ответа, зачем же делали ту или иную фичу.

Причиной возникновения фичи не должны быть аналоги у конкурентов или «мне кажется, будет круто». Без анализа «зачем»

нельзя понять, насколько реализация необходима. «Повысить Retention», «увеличить конверсию в платящих игроков», «снизить процент отваливающихся через неделю игроков» – хорошие цели, основанные на конкретных метриках. Еще лучше, когда есть объяснение, почему выбрано это решение, а не другое, и почему оно должно понравиться разным категориям игроков. Сюда прилагают результаты опросов, аналитики, любые данные, подтверждающие вашу гипотезу. Этот пункт должен предполагать методы проверки, насколько успешной окажется фича после завершения работы над ней.

КОНЦЕПТ – это общее, короткое описание фици без технических подробностей, на основании цели. Концепт должен предлагать конкретные решения возможной проблемы, просто описать ее – недостаточно. Гейм-дизайнеру может казаться, что решение всем очевидно, но это часто не так. Этот пункт нужен, чтобы любой человек смог быстро понять, о чем данный документ; названия для этого не всегда достаточно.

К описанию важно прилагать **РЕФЕРЕНСЫ**. Гифки, видео, картинки помогают понять, о чем идет речь, намного быстрее, чем длинный текст. Когда вы смотрите на готовые решения и видите некоторое количество отличий от собственных предложений, вы можете задуматься, а почему решение реализовано именно так, а не иначе. Ответ на этот вопрос – еще один шаг к проверке собственной идеи.

Можно составить **ЮЗЕР-СТОРИ**, то есть описать, как игрок будет взаимодействовать с каким-то игровым элементом: как узнает о новой опции, как поймет принцип ее работы и взаимодействия с другими элементами, а также, например, как защититься, если игровые соперники применяют новую опцию против него. Описание самой фици здесь минимально, все внимание сосредоточено на ходе мыслей игрока, сталкивающегося с ней, с самой первой встречи. Цель – воссоздать эмоции и цепочку возможных рассуждений игрока, сформулировать желаемую мотивацию. Не просто ввести новую фицу, но понять, почему она станет ценной.

Представляя себя на месте игрока, знакомящегося с фичей, вы увидите и сможете с самого начала подметить и исправить слабые стороны предлагаемого решения. Например, у вас есть класс персонажей, которые могут становиться невидимыми. Как вы объясните другим игрокам, что среди них есть невидимка? Что будет происходить на экране? Отвечая на подобные вопросы, вы получите почти полное описание фичи с точки зрения игрока и его ощущений.

Чтобы юзер-стори была полезной, мало предположить действия игрока, нужно обозначить все, что игрок увидит или почувствует на пути к нужному результату: какие всплывающие окна, какие иконки имеют то или иное значение, что игрок услышал от NPC, чтобы сделать тот или иной вывод. Составлять этот раздел ГДД лучше с помощью блок-схем, наглядно показывающих путь игрока: так вы сразу прикинете архитектуру и объем интерфейсов вашей игры. Юзер-стори может отсутствовать в ГДД, но это хороший инструмент для самого гейм-дизайнера, чтобы еще раз проверить свои идеи.

ЮЗЕР-КЕЙСЫ – это примеры конкретных взаимодействий игрока с фичей. Юзер-кейсы должны покрывать весь предлагаемый функционал. Это описание последовательности игровых изменений, зависящих от действий игрока. Например, нажимая на эту красную кнопку, игрок видит такое-то сообщение (всплывающее окно), в верхнем левом углу появляется таймер; когда время заканчивается, таймер начинает пульсировать; по истечении времени он со взрывом исчезает. Важно описать, как система должна реагировать на все возможные варианты действий игроков, включая негативные, когда игрок делает что-то не по плану гейм-дизайнера.

Если юзер-стори описывает ситуации глазами игрока, то юзер-кейсы описывают происходящее с точки зрения системы. Тестировщик будет проверять, что все работает так, как вы задумали, исходя именно из ваших юзер-кейсов. Грамотно составленный список юзер-кейсов не позволяет QA^[61] пропустить какую-то проблему функционала.

Раздел **КОНФЛИКТЫ** нужен, чтобы предусмотреть, как создаваемая фиша или ее часть взаимодействуют с другими игровыми системами. Такая работа дает возможность заранее увидеть потенциальные конфликты и конкретные шаги по устранению проблем взаимодействия. В разделе, посвященном игровому балансу, мы детальнее рассмотрим примеры, когда отдельные элементы игры, будь то просто предметы экипировки или даже целые фиши, могут вступать в конфликт друг с другом. Такая ситуация часто приводит к дисбалансу по Гарфилду, о котором мы поговорим позже. Пока же просто представим себе ситуацию, что в игре есть ресурсы, один из которых – нефть. Это может быть, к примеру, тактическая онлайн-стратегия. Если на этом ресурсе базируется сразу несколько механик, они неизбежно будут конкурировать за него в голове игрока. Если использование обеих механик не обязательно для игрового процесса, то конфликта нет. В противном случае игрок будет находиться в ситуации неполноценности, пока не прокачает обе фиши. Стоит понимать, что единый ресурс для многих механик не является проблемой сам по себе. В целом это хорошая идея. Конфликт может быть создан конкретной реализацией.

В некоторых компаниях добавляют разделы с читами для QA, мокапами интерфейсов, настройками админки и пр. Все зависит от конкретного проекта и его нужд.

Когда вам не нужно ни с кем ничего обсуждать, вы вольны выбирать любой вариант ведения документации, можете записывать хоть в блокноте. Но, так или иначе, где-то все решения должны быть зафиксированы. Технические инструменты ведения документации помогают собирать воедино части проекта, над которым трудится команда. Где бы вы ни вели документацию, информация там должна быть структурирована. Описательная часть обычно отделена от технической, а последняя имеет подразделы (информация для программистов, QA, аналитиков и т. д.).

Не забывайте также, что ГДД будет наполнен комментариями коллег. Это нормальная практика, когда члены команды могут обменяться мнениями о новых игровых возможностях, чтобы в случае необходимости подкорректировать ту или иную фичу.

ИНСТРУМЕНТЫ УПРАВЛЕНИЯ

Инструментов управления проектами довольно много, вы можете выбирать любые, но это в любом случае будет связка: таск-трекер – база знаний – средство коммуникации.

Таск-трекеры сильно облегчают работу всех сотрудников: вы сразу видите приоритизированные задачи на сегодня, и вам не надо ни к кому подходить, чтобы уточнить, чем лучше заняться.

Если в вашей команде есть человек, ответственный за управление командой, **JIRA** – самый популярный в игровых компаниях инструмент. Наиболее эффективно Jira работает с плагинами, но почти все они – платные. Кроме того, Jira сама платная после достижения определенного количества пользователей. Если команда небольшая, вполне можно обойтись простым и бесплатным Trello.

TRELLO – очень популярное решение для небольших (до восьми человек) команд. Классическая модель предлагает три колонки: список задач, задачи в работе и выполненные задачи: To Do, In Progress, Done. Можно завести отдельную доску под бэклог, чтобы собирать там все идеи по проекту, а после планирования спринта перекидывать выбранные строки в To Do.

Можно настроить, кто имеет право двигать задачи между колонками. Обычно только Scrum Master, если такой есть в команде, двигает задачи из бэклога в список того, что нужно сделать. Человек, который берет ответственность за какую-то задачу, лично перетаскивает ее в раздел «в работе», а Scrum Master переносит ее в «выполнено», когда убедится, что результат соответствует ожиданиям.

CONFLUENCE – это база знаний, открытая для совместного доступа к документам и их редактированию. Если вы используете Jira, оно станет для вас удобным инструментом, так как они интегрированы. Если вы не рассматриваете для себя платный софт, вполне нормально пользоваться Trello и **GOOGLE-ДОКУМЕНТАМИ**.

Нередко встречается, что документация по всему проекту ведется в одном огромном doc-файле. В этом случае лучше сделать отдельную страничку, содержащую ссылки с описаниями на все остальные части файла, чтобы любой член вашей команды мог легко найти нужную информацию. Записывайте краткое резюме всех встреч и звонков и сделайте для этого отдельный документ (взять на себя эту обязанность может Scrum Master).

Последнее, что вам понадобится, – это какой-либо мессенджер. Выбирайте любой привычный. Хочется отметить здесь **SLACK**, так как он интегрирован с Trello и имеет множество полезных функций: например, есть возможность заводить отдельные каналы для разных обсуждений, здесь удобная система тегов по группам, «напоминалки», текст сообщений можно сразу конвертировать в задачу и многое другое.

ТЕХНИЧЕСКИЕ ЗАДАНИЯ

Техническое задание (ТЗ) – это детальная инструкция для исполнителя. Основная цель документа – дать исполнителю понять, что он должен сделать. После этого ему может быть поставлена задача дать предварительную оценку фичи, так что ТЗ помогает спланировать производство, а также рассчитать ресурсы и бюджет на отдельные фичи. На протяжении всего этапа продакшен ТЗ могут корректироваться, это абсолютно нормально, но любые изменения стоит вносить только по договоренности с исполнителем.

В зависимости от особенностей студии и конкретного проекта, гейм-дизайнер может работать с разными исполнителями, готовя

для них ТЗ.

ГДД, описывая желаемый результат, может включать в себя несколько ТЗ, каждое из которых говорит о том, какую именно работу нужно сделать. ТЗ не дает аргументации, зачем, например, нужно добавить какой-то параметр или настроить дополнительную проверку для определения конкретной категории игроков. Программист получает четкую инструкцию, его уже не нужно убеждать в том, что это изменение необходимо. Это верно для всех исполнителей: художников, тестировщиков, UI, композиторов и пр.

Обычно ТЗ ставится ведущим специалистам направления, а те уже распределяют задачи между своими сотрудниками. Гейм-дизайнер редко может лучше поставить задачу, например на написание кода, чем руководитель программистов.

Но бывает, что гейм-дизайнеру нужно ставить ТЗ непосредственно исполнителям, и здесь есть свои нюансы. Например, ставя задачу на тестирование бага, вам необходимо приложить конкретный путь воспроизведения, то есть набор действий, который приводит к проблемной ситуации.

Любой ГДД в итоге должен превратиться в набор конкретных ТЗ для разного рода исполнителей. При этом в ТЗ, как правило, оставляют ссылку на ГДД, чтобы, если это необходимо, всегда была возможность восстановить логику создания фичи. Хотя рассчитывать, что каждый исполнитель, помимо ТЗ, будет каждый раз изучать еще и весь ГДД, довольно наивно. Если вам важен какой-то нюанс, лучше указать на него непосредственно в блоке для конкретного исполнителя.

Идеальной документации не бывает. Для каждой команды, проекта, игрового жанра логично создавать свои шаблоны документов, которые с опытом будут получаться все лучше.

Контроль версий

Если без Confluence или Jira вполне можно обойтись, то без системы контроля версий работать будет очень тяжело, так как при разработке любого продукта, в том числе и игрового, есть необходимость совместной работы над одной общей системой. Задача этой главы – познакомить неспециалистов с системами контроля версий. Программистам эта информация может показаться очевидной и упрощенной.

На поиски маленькой ошибки в коде игры могут уходить часы работы. Системы контроля версий позволяют всем участникам отслеживать выполнение задач другими специалистами, легко вносить изменения и в случае, если что-то пошло не так, иметь возможность откатить эти изменения. Последнее делает оправданной работу с системами контроля версий даже для инди, работающего в одиночку. Ведь хранилища данных, тот же Dropbox, например, накладывают ограничения на возможность вернуться к нужной версии.

Базовый вариант, когда все данные хранятся на сетевом диске, предполагает, что внесенные изменения заменяют предыдущую версию, так что можно легко стереть чужую работу. Ключевая особенность систем контроля версий: вы обмениваетесь не целыми файлами (картинками, частями кода и пр.), а непосредственно изменениями имеющихся. Поэтому люди могут работать над одним проектом, видеть историю всех изменений и легко возвращаться к нужному состоянию.

Систем контроля версий довольно много. Самые известные: Git, SVN, Perforce, Mercurial и др. В отличие от смены игрового движка, перейти с одной системы контроля версий на другую несложно.

SVN выбирают обычно в случае, когда важна максимальная простота вхождения. Perforce редко используется в России, но он отлично подходит для крупных проектов, над которыми будут

работать удаленные сотрудники. Большие компании нередко пользуются этим закрытым платным решением, нанимая дешевых исполнителей за рубежом, ведь Perforce быстро синхронизируется на удаленных компьютерах. Но большинство разработчиков выбирают бесплатный Git, позволяющий быстро и удобно работать с данными.

Рассмотрим подробнее SVN и Git.

SVN – это две сущности: репозиторий (хранилище данных), который лежит на удаленном сервере, и рабочая (локальная) копия проекта, с которой взаимодействует сотрудник.

Для простого пользователя, не администратора, процесс работы с SVN выглядит довольно просто. Сначала необходимо установить клиент этой системы на свой компьютер. После чего, зная адрес репозитория, у пользователя будет возможность получить в рабочую копию нужные ему файлы. Внеся изменения, например, в код игры, программист сможет отправить результаты своей работы на удаленный сервер.

Если операция проходит успешно, его работа сохраняется на удаленном сервере, и теперь все другие сотрудники смогут забрать новую версию игры с внесенными изменениями.

В некоторых случаях система может не принять работу. Например, возникли технические проблемы на сервере или у сотрудника нет прав вносить изменения в конкретное место и т. д.

Могут случаться и конфликты, если над одним файлом одновременно работал кто-то еще и внес противоречащие изменения. В этом случае можно отменить свою работу, пересекающуюся с работой другого специалиста, либо, наоборот, признать свою работу приоритетной, либо же открыть файл и вручную, осознанно править его. Художники обычно работают над своим искусством самостоятельно, поэтому это более актуально для программистов и гейм-дизайнеров, выполняющих общие задачи.

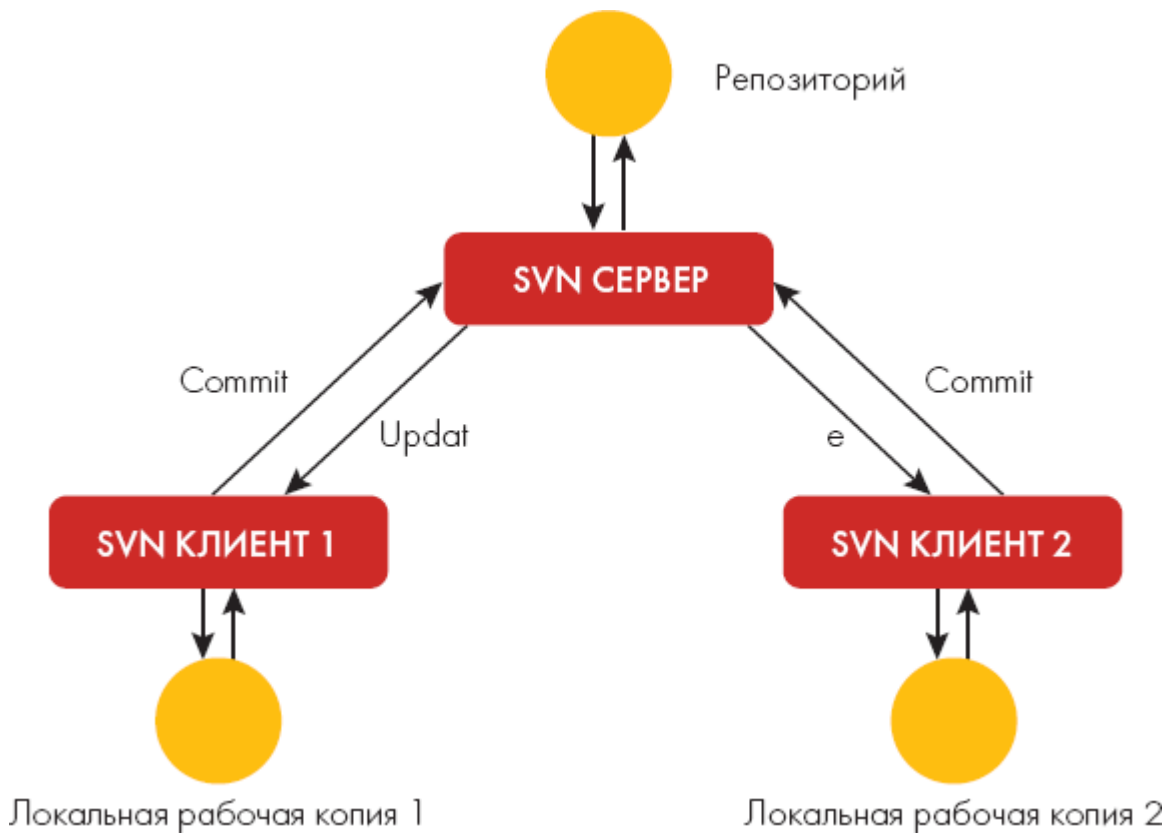


Рис. 16. Схема SVN

Commit – сохранение, фиксация (в архиве, репозитории и др.) изменений в программном коде.
Update – сверка текущей версии кода у сотрудника с версией в репозитории и обновление её в случае наличия более новых файлов на сервере.

Работа с **GIT** выглядит так: есть удаленный репозиторий, локальный репозиторий и рабочая копия. Это значит, что появляется один дополнительный по сравнению с SVN шаг – синхронизация локального репозитория с удаленным. Это дает большое преимущество – возможность работать сразу над несколькими задачами. Пока изменения, которые вы вносите, находятся в промежуточном состоянии и могут «сломать» версию, они сохраняются в локальном репозитории, не создавая проблем остальным. Так что сотрудник может в любой момент вернуться к выполнению задачи.

Допустим, мы работаем над чем-то рискованным: например, решили поменять всю игровую физику. Чтобы не мешать остальным

сотрудникам, имеет смысл создать отдельную ветку. Ветка – это последовательность изменений, параллельная ветка репозитория. Она не влияет на главную версию, так что сотрудники могут работать над своими задачами одновременно, не боясь что-то сломать.

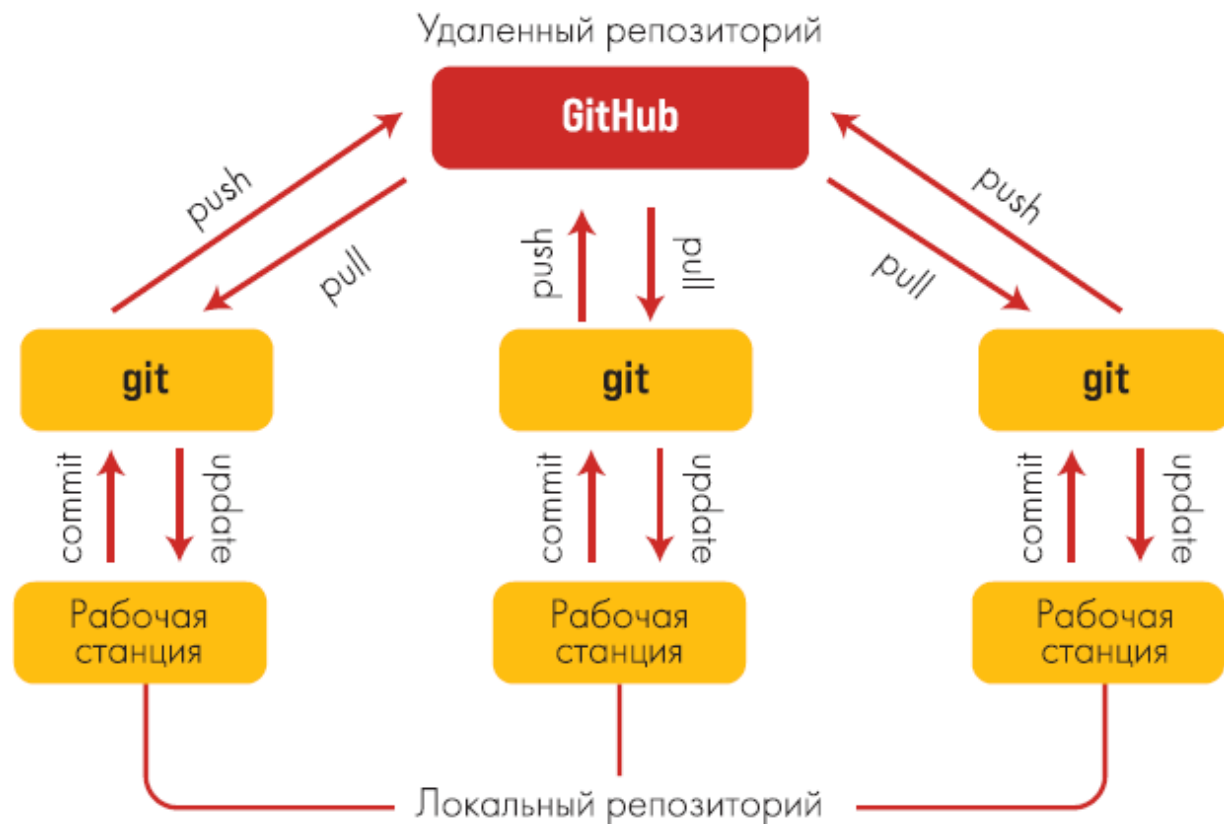


Рис. 17. Схема Git

Дизайнеры и художники могут, например, спокойно работать в основной ветке, а программисты – в своей. Во многих студиях существует правило: «Одна задача – одна ветка», что позволяет экономить время на тестировании вносимых изменений.

В SVN тоже можно создать отдельные ветки, только в общем репозитории. Но Git позволяет работать с ветками гораздо быстрее и удобнее.

Когда все будет протестировано и принято, изменения можно будет внести в основной проект.

Ответвление позволяет также решить вопрос, как показывать билд, например, инвестору или на конференции. Всегда существует вероятность, что какое-то изменение сломает билд, а нам нужно показать работающую игру или ее часть. Вместо того чтобы ничего не делать, боясь что-то испортить (что ведет к простоям и потере времени), просто создают отдельную ветку, где программисты работают только над устранением багов. В то же время их коллеги могут спокойно дальше разрабатывать игру, не рискуя потерей рабочего билда.

Continuous Integration, то есть практика разработки, при которой рабочие копии постоянно сливаются в основную ветку, предполагает, что никто не работает «в стол», а как можно быстрее отдает свою работу в общий проект. При таком подходе версия игры не должна собираться раз в месяц, все должно быть собрано хотя бы за последние пару дней. Живая версия, в которую сотрудники могут поиграть, интегрируется непрерывно, чтобы вовремя отследить возможные проблемы.

Если накатывается много изменений, чтобы версия была стабильна, обычно договариваются о времени отсечки. Скажем, все изменения, внесенные после 17:00, не попадут в завтрашнюю версию. Таким образом, если обнаружатся какие-то проблемы, сегодня еще будет время для их устранения, и все сотрудники на следующий день получат рабочий билд, в который можно играть.

По сути, работа в системе контроля версий для рядовых сотрудников – это несколько операций: забрать из репозитория данные, отправить обратно свои изменения или сменить ветку и, если нужно, посмотреть работу других специалистов. Для новичков все это может выглядеть довольно страшно, но в большинстве случаев хватает рабочего дня, чтобы более или менее разобраться, что к чему, и приступить к работе.

С точки зрения администратора, есть еще одна важная задача: решить, где, собственно, будет храниться репозиторий. Первое решение – личный компьютер. Это самый простой вариант, и его часто используют разработчики, работающие с Git, из-за логики локальных репозиториях, позволяющей никому не платить и работать, даже когда общий удаленный репозиторий по тем или иным причинам недоступен.

Второй вариант: поставить или арендовать отдельный компьютер под сервер, например, того же Git, который будет заниматься только этим; так делают почти все крупные компании. Это безопасно, никто извне не сможет посмотреть или получить информацию, но обычно дороже, чем другие варианты.

Можно разместить репозиторий у сторонней компании, например GitHub. Так как сервер располагается не у вас, у такой компании будет доступ к вашим файлам, но большинство инди-команд это не останавливает. Зато вы перекладываете всю работу по поддержке сервера на внешнюю фирму, что освобождает вашего системного администратора от обязанности следить за его состоянием. Большинство таких сервисов довольно надежные и не допускают у себя проблем, которые могут парализовать вашу работу.

Это решение часто бесплатное, но может налагать некоторые ограничения. Например, бесплатная версия может предполагать только публичное размещение вашего проекта или ограничить количество пользователей/объем данных и пр.

Еще одной базовой вещью при работе с репозиторием является система хуков (hooks). Это скрипты, срабатывающие при определенных событиях и проверяющие входящий коммит – операцию фиксации изменений. Например, мы можем заранее обозначить разрешение для текстур, чтобы исключить возможность ошибки художника. Или запретить создавать ссылки на несуществующие файлы. Крупные студии могут создавать более сложные правила проверки, например, чтобы вносимые изменения не могли нарушить сложные отношения игровых сущностей. Такая

автоматизированная проверка помогает избежать лишней работы по исправлению вручную, ведь всегда проще предотвратить, нежели потом чинить.

Другая проблема заключается в том, что сборка крупного проекта в билд, в который можно поиграть, может быть долгой. Неэффективно, когда каждый сотрудник студии для ознакомления с новой версией игры должен вручную запускать сборку и ждать, пока компьютер обрабатывает ее сорок минут. В этом случае на помощь приходят билдеры – программы, позволяющие непрерывно собирать версию (например, Teamcity). Обычно такой софт бесплатный и несложный в настройке.

И Git, и SVN позволяют настроить интеграцию с различными таск-трекерами. Это помогает сразу видеть, кто, когда и по какой задаче внес те или иные изменения, что сильно облегчает процесс отслеживания работы сотрудников.

Игровой баланс

Какие ассоциации вызывает у вас термин «игровой баланс»? Если вы давно играете в игры, особенно мидкорные и хардкорные, вы, вероятно, вспомните про честность, правила, равные условия для всех игроков и наличие шанса на победу у каждого. Разработчики же обычно смотрят на баланс шире.

Итак, игровой баланс – это равновесие между элементами игрового процесса, объектами и средствами достижения целей. Работа с балансом – не чистая математика, это все еще работа с геймплеем. Важно знать правила своей игры, понимать ход развития событий. Лучше работать над балансом вплотную после утверждения базовых механик, чтобы не пересчитывать все характеристики заново. Только определившись с правилами, логикой и ограничениями, которые лягут в основу игры, есть смысл приступать к расчетам и плейтестам.

Первые цифры, скорее всего, есть уже на этапе идеи. Жанр, платформа и особенности игры обычно сразу позволяют выявить соотношение основных сущностей. Например, мобильные игры из-за размера экрана не позволяют оперировать большими числами; если игроку нужно считать в уме, тоже стоит задуматься об упрощении; чтобы показать ценность эпического предмета перед обычным, разница в их числовых характеристиках должна быть очевидной и т. д.

Еще на этапе создания прототипа были использованы и проанализированы первые пробные значения. Допустим, в игре есть орки и эльфы. Что они должны уметь? Образ может диктовать характеристики: ловкие, но плохо защищенные эльфы, сильные, но медлительные орки. Или наоборот: сначала у вас есть некие способности и характеристики, а образ подбирается уже под них.

Изначально важны не сами цифры, а их соотношение, базирующееся на геймплее. Что будет, если увеличить или

уменьшить эти значения вдвое? Скажется ли это на геймплее? Результаты этих исследований на прототипах – ваши входные данные для дальнейшего расчета баланса. Если продукт еще не готов, можно попробовать в него поиграть хотя бы мысленно, чтобы ответить на вопросы: «Сколько атак в среднем нужно для убийства такого-то монстра?», «На каком уровне игрок способен справиться с боссом?», «Сколько максимально ресурсов может получить игрок за одну игровую сессию?» и пр.

Давайте подумаем, какого баланса вы хотите как игрок? Идеального или не очень? Чтобы это понять, необходимо определить, какой же баланс мы считаем идеальным. К примеру, если в МОБА-игре есть 100 персонажей, то идеальный баланс предполагает равные шансы победы независимо от того, какого персонажа вы выбрали. То есть на бесконечном количестве игр каждый персонаж покажет винрейт [\[62\]](#) 50 %.

Чтобы убедиться, что мы создали именно таких персонажей, можно запустить автотесты с ботами, которые сыграют очень много матчей и соберут нам статистику. После нескольких итераций, правок и тестирований разработчику удастся добиться, чтобы винрейт у каждого персонажа с заданной точностью стремился к 50 %.

И вот долгожданные герои попадают к игрокам. И какие же отзывы получает игра? Баланса нет, уйду в «Доту»! Статистика подтверждает негативные комментарии. Один персонаж показал значительно лучшие шансы на победу, другой непопулярен, несмотря на высокий винрейт, а третий настолько часто проигрывает, что совершенно никому не нужен. Гейм-дизайнеры начинают работу над ошибками. Оказалось, что интересный лор и яркий визуал увеличил используемость некоторых героев, и они в среднем имеют более низкий винрейт, так как за них играет больше новичков. Другие герои стали популярны лишь в умелых руках, а третьи просто выпали из игры. Да и автобот, тестирующий стратегии, был написан неидеально: например, пользовался самыми

базовыми геймплейными возможностями и при этом не допускал ошибок, свойственных игрокам. Игроки придумывают все новые и новые тактики, которые никак не удается сбалансировать.

Можно ли хоть как-то учесть разные навыки игроков? Тысячи тысяч комбинаций? Сочетания способностей и слабостей героев? Физики сравнили бы эту ситуацию с детерминированным хаосом. Это понятие означает, что наша система кажется хаотичной, она живет по своим законам. Но на самом деле за всем этим стоит логика, формулы, правила. В теории мы можем просчитать абсолютно все идеально! Все варианты. Ведь все формулы известны: мы сами их составляем. Все факторы влияния рано или поздно будут найдены. Вот только сложность такой задачи крайне велика. Количество переменных и уравнений чуть ли не бесконечно. И вроде бы кажется, что мы можем просчитать все, но на практике нет таких гейм-дизайнеров, как и не существует таких вычислительных мощностей. Плюс добавление новых персонажей пошатнет этот, таким трудом достигнутый, баланс. Стоит ли вообще тогда пробовать решить эту задачу?

В этот момент мы возвращаемся к тому, с чего начали. К понятию баланса. Чтобы сделать задачу решаемой, давайте сформулируем его точнее. Поймем, что же на самом деле мы хотим от «игрового баланса».

Пойдем от обратного. Будем считать, что идеальный баланс – это ситуация, при которой нет дисбаланса. Соответственно, хороший баланс – это когда есть дисбаланс в рамках некоего допустимого уровня. Когда игрок говорит, что у вас дисбаланс, он выражает какие-то эмоции. И эти эмоции часто не в пользу проекта. Он недоволен! Тем, что он проиграл (возможно, чисто по своей вине в равном и честном бою). Тем, что условия были неравные: персонаж не подходящий или противник заведомо сильнее. Или тем, что игра скучная и не вызывает никаких эмоций. Психологи проводили исследование американских студентов, попросив их оценить себя и отнести к одной из групп по уровню оценок, социальному статусу и

т. д. Сколько бы ни повторялось исследование, результат был одинаков: люди слегка завышали свое положение. Абсолютное большинство считало себя выше среднего. Применяя эти знания в играх, мы можем выдвинуть гипотезу, что игроки окажутся недовольными, даже если их винрейт будет 50/50. Они хотят побеждать хотя бы чуточку чаще, чем средний игрок. Но PvP-сражения – это игра с нулевой суммой. Если одна команда одержала победу, другая определенно проиграла. Другого не дано. Получается, даже идеальный баланс с винрейтом 50 % оставит игроков недовольными?

Все это приводит к размышлениям, а нужен ли идеальный баланс вообще? Или точнее, имеет ли этот термин хоть какой-то практический смысл? Баланс тетриса более совершенен, чем у MMORPG, в которой есть десяток классов персонажей с большим количеством разных способностей. Но делает ли это тетрис более глубоким, интересным и реиграбельным? Что ж, здесь мы вплотную подошли к целям игрового баланса.

ЦЕЛИ И ОСОБЕННОСТИ БАЛАНСИРОВКИ ИГР

У всего, что мы делаем на работе, должна быть цель. Каждый раз мы задумываемся, почему в игре должно быть именно 36 персонажей, 18 локаций и 60 уровней. Это требование продиктовано скоростью производства контента, его минимальным объемом для функционирования игрового цикла или соображением создания нужного Retention (удержания игроков)? И так каждый раз, когда мы начинаем любую работу. Так чего мы хотим добиться тем самым балансом? Выделим несколько целей:

- ощущение честности;
- повышение реиграбельности;
- контроль сложности;
- стимулы к монетизации.

Разумеется, это не все. В каждой игре разработчики ставят свои цели.

Балансировка игры – это возможность сравнить параметры игровых сущностей и в заданных условиях влиять на эти характеристики. В отличие от технических задач, создание игрового баланса – вещь многогранная, подходить к которой можно по-разному. Для одной игры критически важны математические расчеты, без которых велик риск допустить преимущество одного класса персонажей перед другим; для другого проекта это может иметь меньшее значение. Зато гейм-дизайнерам необходимо, например, понимать, сколько времени человек должен играть, чтобы достичь максимального уровня.

Подходы и методы расчета игрового баланса могут отличаться, если игра полностью заточена под **PVP ИЛИ PVE – РЕЖИМЫ**^[63]. Для первого важно сохранить близкие к равным условия для всех игроков. Возьмем *Mortal Kombat*: мощь не влияет на исход драки, только навык и рандом. Ты никогда не знаешь, как станет играть твой оппонент, поэтому каждый бой будет не похож на предыдущий. Игра с реальным человеком уже сама по себе добавляет фана. В PVP больше требований к балансу, зато меньше контента, и в каком-то смысле разрабатывать его даже проще.

ДЛЯ ОДНОПОЛЬЗОВАТЕЛЬСКИХ ИГР на первый план выходит необходимость так сбалансировать уровни, чтобы чувствовался вызов: не было ни скучно, ни утомительно. Сложность игрового процесса должна прогрессировать вместе с навыками играющего, плюс она обычно совпадает с драматургией повествования.

Это особый вид баланса, и это же одна из целей – построить **КРИВУЮ СЛОЖНОСТИ** так, чтобы игроки не уходили из-за слишком простых уровней или слишком сложных. Обычно за этим стоит сложная математика, но мы попробуем рассмотреть ее на упрощенном примере.

Сложность в общем случае – это некая величина, характеризующая текущий уровень давления на игрока

относительно его позиции в игровом прогрессе. В тех же три-в-ряд прогресс игрока определяется уровнем в цепочке, до которого он дошел. Осталось численно определить степень давления на игрока. К примеру, это может быть влияние на сложность доступного игроку числа ходов, механик и геометрии уровня. А может быть процент успешных попыток пройти уровень к общему числу попыток. Или с какого раза игрок в среднем проходит текущий уровень. Что ж, теперь мы знаем достаточно, чтобы сформулировать следующий тезис:

Создание игрового баланса – это искусство выразить необходимый уровень эмоций через числовые значения.

Некоторые игры позволяют игроку самому выбирать уровень сложности. Другие предполагают искусственное завышение мощности мобов, чтобы противники всегда повышали свои характеристики раньше, чем игрок. Некоторые гейм-дизайнеры предусматривают варианты, когда сложность подстраивается под действия игрока или убавляется только там, где игрок систематически не справляется. Нередко в игре есть варианты активностей, чтобы игрок по настроению мог выбирать себе более простое или сложное занятие: побочные квесты, сбор ресурсов, сражения в подземельях и так далее.

СИММЕТРИЧНЫЕ МНОГОПОЛЬЗОВАТЕЛЬСКИЕ ИГРЫ предполагают, что каждый человек начинает играть в равных условиях и имеет одинаковый набор стратегий победы. Такие симметричные игры тоже должны быть интересными и разнообразными, так что здесь гейм-дизайнеру предстоит работа над балансом внутриигровых предметов и сущностей. Например, сражаться номинально разными, но дающими один и тот же эффект мечами – скучно. Чтобы разнообразить геймплей, гейм-дизайнеры добавляют разные характеристики и возможности взаимодействия, и необходимо рассчитать, как они между собой соотносятся.

Такое соотношение стоимости и преимуществ называют **КРИВОЙ СТОИМОСТИ**. Она может быть линейной (чем дороже, тем лучше)

или может зависеть от других факторов, например когда каждое следующее преимущество становится дороже.

У игровых объектов может не быть прямого соотношения стоимости и преимуществ. В этом случае можно балансировать их по аналогии с игрой «камень, ножницы, бумага», когда ни один из вариантов не является выигрышным, так как первые сущности по умолчанию сильнее вторых, но проигрывают третьим. Такое решение часто встречается в классовой системе, когда воин, например, сильнее лучника, но слабее мага. Невозможность победить противоположный класс и ограничение выбора игрока можно нивелировать, повторив такое деление внутри каждого узла. Например, воинов поделить на варваров, рыцарей и разбойников, магов – на священников, некромантов и знахарей и т. д., при этом опять же дать каждому классу как преимущество, так и недостаток. Вариантов становится намного больше, сочетания разных способностей дают разнообразие геймплея. Но с каждым новым классом математически считать их баланс становится все тяжелее.

Игры **АСИММЕТРИЧНЫЕ, С НЕРАВНЫМИ ИЗНАЧАЛЬНЫМИ УСЛОВИЯМИ**, балансировать сложнее. Некоторые игры намеренно предлагают игрокам неравные условия: например, чтобы передать ощущение несправедливости жизни или показать уникальные механики выбранной расы, как в *Starcraft*.

Удобно, если сущности можно навсегда привязать друг к другу: например, один человек начинает игру с одной единицей древесины, а другой – с двумя камнями, и эти ресурсы всегда соотносятся друг с другом один к двум. Если же в вашей игре персонажи имеют разный доступ к ресурсам, разные возможности взаимодействия с миром, разные цели и стратегии, потребуется намного больше плейтестов.

Гейм-дизайнер должен продумать, какая из игровых сущностей окажется сильнее при столкновении, – неважно, идет ли речь о двух игроках, об армиях персонажей, о наборах карточек, скорости

машины и пр. Ваша задача – уметь их сравнивать, остальные вопросы касаются уже визуализации.

Для этого прежде всего необходимо выписать все сущности, имеющиеся в вашей игре. Чтобы их сравнить, нужно привести все к цифровому формату, не забыв прописать, как они влияют на геймплей. В любой игре есть важные числовые параметры, будь то здоровье (HP) персонажа или угол подачи мяча в футболе.

Получив список с числовыми значениями суммарной мощи (суммой всех характеристик), предстоит решить, как эти характеристики будут меняться (то есть определиться с формулами прокачки, если она нужна), а также по каким формулам и законам эти сущности соотносятся между собой. Главным инструментом для расчетов остается Excel, для более сложных формул используют Wolfram Research и другие программы.

Итак, **ЧТО ЖЕ МЫ БАЛАНСИРУЕМ?**

Как мы уже выяснили, баланс – это своего рода выражение эмоций через цифры. Эмоции в свою очередь – это вся игра. Мы играем, потому что нам нравится испытывать те или иные ощущения от игрового процесса.

В свою очередь игра – это игровой цикл, набор механик и дополнительных фишек. Получается, что игровой баланс – это баланс всех этих сущностей, как по отдельности внутри каждой, так и между собой.

С точки зрения подходов и тех элементов, которые мы балансируем, можно выделить множество игровых аспектов, каждый из которых требует своего подхода и имеет свой вид баланса. Давайте приведем несколько примеров.

- Баланс начальных условий

Это про то, чтобы все стартовали в более или менее равных условиях. В MMORPG это равнозначность стартовых локаций и первых способностей всех рас. В MOBA/action – подбор игроков, близких по навыку. И так далее.

- Баланс сложности прохождения

Он касается как прохождения уровней в казуальных играх или локаций и подземелий в многопользовательских играх, так и в целом прохождения однопользовательских игр. К примеру, выбор уровня сложности в *Diablo* или «*Цивилизации*».

- Баланс мощи/случайности/навыка

Обычно касается балансировки сражений в многопользовательских играх. Как с помощью рейтинга Эло^[64] определить навык игрока? Как подобрать адекватных ему по силе противников? А что делать, если экипировка или монетизация влияют на его силу?

Продолжать этот список можно очень долго. В каждой конкретной игре он будет свой. Помимо видов баланса, выделяют также баланс конкретных фичей, присутствующих в разрабатываемой игре.

Это баланс экономики, прокачки и развития сущностей, общий баланс всего игрового цикла и то, что многие обычно и понимают под балансом, – боевая система. Давайте остановимся на ней подробнее.

ПРОЕКТИРОВАНИЕ БОЕВОЙ СИСТЕМЫ

Начиная разработку любой фицы, любой системы или игровой механики, необходимо задуматься, что мы делаем и какие цели преследуем. Есть два подхода: когда мы знаем, что хотим увидеть в конце, и когда не знаем. В первом случае работа представляет собой декомпозицию конечной цели на промежуточные шаги, реализация которых приведет к выполнению задачи. Во втором мы занимаемся исследованием или прототипированием, чтобы получить механику, которая даст нужные нам ощущения. К какому же типу относится боевая система?

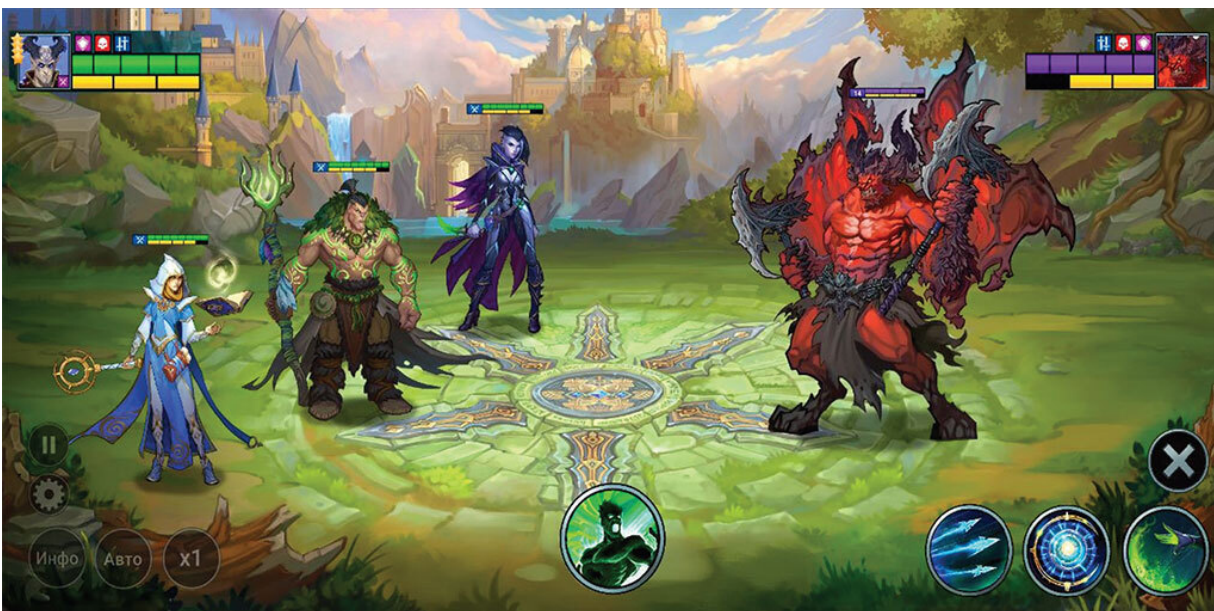


Рис. 18. Бой в игре

Чтобы ответить на этот вопрос, давайте определимся с тем, а что же такое эта «боевка». Если между собой сражаются два игрока, персонажа, войска или государства, нам интересно узнать, кто победит. Поэтому самое простое определение боя – это применение способностей сражающихся субъектов, обычно нацеленное на достижение победы и визуализируемое сражением. Это применение может быть как автоматическим, так и контролируемым. Каждый игрок может иметь как одну способность (атака), так и целое многообразие возможностей. С математической точки зрения боевая система – это механика, подразумевающая сравнение боевой силы, которая позволяет игрокам узнавать, кто из них одержит победу. Во многих играх сражение представляет собой активный геймплей. Нередко это даже core-геймплей всей игры. В других играх, например в батлерах, мета^[65] важнее боя.

Оставим на потом вероятность победы, расчет потерь и другие нюансы. Для начала вспомним: чтобы сравнить между собой две сущности, необходимо выразить их численно. Гейм-дизайнеру нужны параметры, по которым он будет сравнивать. Кто круче: титан или черный дракон? Для людей ощущение крутости, как и многие

другие чувства, работает по логарифмической шкале. Это позволяет нам слышать писк комара и неглохнуть от самолетного гула. Вспоминая «Героев Меча и Магии», вы наверняка скажете – конечно, драконы сильнее титанов! А во вселенной *Warcraft* даже черный дракон Смертокрыл был лишь одним из многих слуг титанов. Наши чувства относительны. Но компьютер требует точных цифр. Давайте же их введем.

Для игр, где есть боевая система, главные характеристики – урон (англ. damage) и HP. К этим характеристикам можно привести многие способности, например любой бафф (временное усиление игрока) – это увеличение либо урона, либо количества жизни юнита; замедление приводит к уменьшению урона, так как за единицу времени игрок сможет ударить меньшее количество раз, и т. д.

Обозначим за P величину, характеризующую силу и крутость сражающихся существ. Назовем ее, например, мощностью. Тогда существо, обладающее мощностью P_1 , большей, чем мощность P_2 второго существа, в среднем будет одерживать над ним победу в бесконечном числе сражений. В некоторых играх на этом все и заканчивается. Но большинство из нас скажут: подождите, это же очень скучно! И тут на помощь приходят системы характеристик. Гейм-дизайнеры наделяют существ ловкостью, интеллектом, выносливостью и множеством других атрибутов. Для конкретной задачи можно использовать как устоявшуюся систему – из D&D, GURPS, SPECIAL, систему из *Diablo* и т. д., – так и разработанную самостоятельно. Про каждую из приведенных в пример систем характеристик можно найти массу информации в Сети. По сути, они представляют собой набор параметров персонажа и правил, по которым они влияют на ход боя. Какую бы систему вы ни взяли, за ней будет стоять математическая модель. Если говорить очень грубо, ее задача – свести все параметры сравниваемых объектов к их мощности. Например, одна единица силы может давать +1 к физическому урону и +5 к запасу здоровья. Задумываясь о боевом

процессе глубже, мы приходим к выводу, что все характеристики можно привести к урону (или урону в секунду) и запасу здоровья.

Какой персонаж победит? Очевидно, что тот, кто успеет довести до нуля здоровье противника до того, как противник сделает с ним то же самое. Тогда, обозначив за D урон, а за H – запас здоровья, мы можем записать условие победы первого игрока над вторым:

$$H1/D2 > H2/D1$$

Вспоминая, что и урон, и запас здоровья, скорее всего, больше нуля, перенесем $D1$ и $D2$ в противоположную сторону неравенства:

$$H1*D1 > H2*D2$$

Но подождите! Мы же начинали с того, что если какая-то обобщенная характеристика игрока больше той же характеристики другого, то первый, скорее всего, побеждает второго. Так мы и пришли к простой формуле мощи:

$$P = H*D$$

Используемые в этой формуле значения урона и запаса здоровья часто называют эффективными или приведенными, потому что они сами рассчитаны из других характеристик персонажа: интеллекта, ловкости и т. д. К примеру, у персонажа с $HP0 = 30$ и сопротивлением урону 50 % эффективное HP будет 60 (то есть $HP0 / 50 \%$). Таким образом, получив урон 30 (равный первоначальному HP), он потеряет только половину здоровья (15 из 30).

Конечно, представленная модель очень упрощенная, и в реальности все может быть гораздо сложнее. Но нам важно показать логику рассуждений и как вообще можно подойти к проработке собственной боевки. Реальные цифры подгоняются на основании плейтестов.

Есть способности, которые нельзя привести к одному показателю. Например, они могут быть слишком разными либо на первый план выходит навык игрока, а не численные характеристики объектов. Как математически сравнить возможность кидать фаерболы и телепортироваться? Сбалансировать такие вещи можно только с помощью плейтестов.

Рандом (случайность) – тоже инструмент баланса, и многие гейм-дизайнеры им пользуются. Элементы рандома вносят разнообразие и азарт, что добавляет остроты ощущений.

Мощь, навык игрока и рандом – три составляющие игрового баланса, влияющие на победу. Такой подход позволяет слабым игрокам побеждать сильных, не дает последним скучать и расслабляться. Боевые системы по-разному учитывают комбинацию из этих трех сущностей.

Hearthstone – игра, в которой почти идеально сочетаются параметры мощи, рандома и навыка. Здесь играет роль как сила выбранных карт, так и знание игроками разных колод (умение предугадать действия противника позволяет навязать свою тактику боя). В то же время каждая партия уникальна, так как оппонент подбирается случайным образом, а карты из колоды выпадают рандомно.

Если у вашей команды мало опыта, самостоятельно рассчитать игровой баланс и построить рабочую боевую систему будет непросто; имеет смысл обратиться к похожим играм. Перебирать движок чужой игры может оказаться тяжело, но у больших игр есть Wiki, где вы можете отыскать формулы расчетов. Даже если вы не хотите использовать готовые решения, стоит с ними ознакомиться, чтобы было от чего оттолкнуться. При анализе чужой игры тоже следует выписывать значения характеристик, соответствующие, например, уровням персонажа. Так вы сможете определить основные переменные и их зависимость друг от друга.

ДИСБАЛАНС

Иногда гейм-дизайнеры намеренно вводят дисбаланс, чтобы за счет большего количества стратегий победы сделать игру более интересной. Представьте, что вам на последнем ходу удалось победить многократно превосходящего по силе противника, – именно такие игровые моменты вызывают сильные эмоции, их любят стримеры и обозреватели.

Намеренный дисбаланс позволяет уравнивать шансы компьютера и игрока или же дает заведомо слабым игрокам преимущества на старте.

Для части игр, где все должны быть более или менее равны, это может оказаться проблемой, для других – добавит разнообразия в геймплей. Стратегические игры нередко перерабатывают баланс юнитов уже после релиза, – это провоцирует игроков на поиск новых стратегий, приносящих интересный опыт.

Гейм-дизайнеры иногда используют термин «дисбаланс по Гарфилду». Он означает бесполезность или гиперполезность отдельных элементов игры. Один из авторов рассуждений на эту тему – Ричард Гарфилд, известный гейм-дизайнер, на счету которого такие карточные игры, как *Star Wars TCG* и *DOTA Artefact*. Но главное творение Ричарда стало мировой классикой: *Magic: The Gathering*.

ПЛЕЙТЕСТЫ И БАЛАНС

К сожалению, универсальной методологии не существует: идеальная математика не гарантирует, что на практике не вскроются несоответствия. Поэтому необходимы плейтесты. Допустим, что по цифрам все сбалансировано, наш персонаж наносит самый большой урон и имеет наибольшее количество жизни, но в реальности все равно постоянно проигрывает. Почему? Плейтесты могут показать,

что, например, ему не хватает скорости и он просто не успевает добежать до оппонентов. Возможные результаты таких проверок – корректировка не только цифр, но и механик: например, нашему герою можно добавить способность замедлять противников.

Плейтесты также помогут определить «сложность» вашей игры. Если пока у вас небольшой опыт гейм-дизайнера, то правильно оценить ожидания аудитории может быть нелегко. На самом деле, если аудитория неоднородна, игра в любом случае окажется слишком простой для одних игроков и слишком сложной для других. Вы можете добавлять/менять правила и условия, за счет чего разделять сложность прохождения, чтобы угодить разным категориям игроков, или же ориентироваться на средние показатели, чтобы охватить самую широкую аудиторию.

Во время тестов стоит обратить внимание также на самые популярные решения игроков. Может быть, они покупают какой-то артефакт чаще всех остальных или же никогда не выбирают какое-то действие, хотя, по вашему мнению, оно должно быстрее привести к победе; анализ таких ситуаций поможет избежать появления одной выигрышной стратегии. Для казуальных игр в этом нет ничего особенного, но если игроки пришли к вам за вызовом, то, осознав, что для выигрыша достаточно каждый раз делать одно и то же, они могут разочароваться. Если игра предусматривает одну выигрышную стратегию, подумайте, нужно ли давать игрокам возможность следовать другим, «ложным» стратегиям и для чего вы вводите это разнообразие.

Плейтесты без проработанных математических моделей тоже не смогут проверить все возможные ситуации. Так что сначала делаем расчеты, после чего максимально большой командой проверяем, не расходится ли теория с практикой. Но чтобы наигранные вами или игроками тысячи сессий предоставили материал для размышления, необходимо подключить сбор статистических данных. Для этого существуют различные внешние трекинговые системы, позволяющие расставить в проекте триггеры, срабатывание которых будет

отправлять те или иные данные на сервер. Например, полезно отслеживать каждый обмен ресурсом, чтобы проверять экономику, записывать состав команд в каждом бою, его продолжительность и результаты. А иногда даже фиксировать все действия игроков во время боя. Команда разработки может написать свою систему для сбора и визуализации статистики. Но в современных реалиях мы имеем множество отлично заточенных под игры систем аналитики, и создать собственное решение и использовать именно его – разумный шаг только для очень больших компаний, ведь, помимо разработки, такая система будет требовать постоянных затрат на поддержание.

И помните, что никто лучше гейм-дизайнера не знает, как должна работать та или иная фишка. Поэтому даже лучший в мире отдел QA не заменит гейм-дизайнерского тестирования. Играйте в свои игры!

Игровая экономика

Еще одно многогранное понятие, тесно связанное с игровым балансом и монетизацией, – игровая экономика. Определим ее как совокупность всех изменений характеристик численных игровых параметров, зависящих от времени, взаимодействия с игровым миром, действий других игроков и сервисов оперирования проекта.

Как быстро игрок получает награду в рейде – пример неденежной экономики, которую задает сам разработчик; скидку на «Набор новичка» во время акции обычно определяет уже тот, кто занимается оперированием готового проекта. Мы сосредоточимся на первом варианте, где цель игровой экономики – развлекать игрока, а не монетизировать его. Хотя нередко эти вещи тесно переплетены, особенно во free-to-play играх.

Примером игровой экономики может являться не только количество золота, которое игрок получает за квест, или стоимость оружия в игровом магазине. Прокачка по опыту тоже входит в экономику, если мы решили, что, допустим, каждые три уровня игрок должен покупать новое оружие, чтобы быть в состоянии убивать новых монстров. Здесь денежные показатели напрямую связаны с получением опыта, так что нельзя задавать эти параметры отдельно.

В игре существует некий поток ресурсов. С точки зрения их оборота экономика может быть закрытой, когда человек взаимодействует только с самой игрой, или открытой, когда игроки могут обмениваться ресурсами между собой.

Если рассматривать **ЗАКРЫТУЮ ЭКОНОМИКУ**, в теории все относительно просто: ваша задача – уравнивать общую сумму всех входящих ресурсов и общую сумму их расхода. Игрок может получить ресурсы за активные действия (например, убийство моба) или пассивно (рудник дает в день 2 руды).

Закрытая экономика – негибкая, ее сложно сбалансировать для всех категорий игроков. Она не учитывает разные навыки

конкретных пользователей, и, как следствие, особое внимание придется уделить кривой стоимости, тщательно просчитав все за игрока. Открытая экономика, в свою очередь, позволяет игрокам самим влиять на рынок ресурсов.

Чтобы сбалансировать закрытую экономику, гейм-дизайнеры вводят понятие общего «уровня» или «мощи», от которого строится баланс всех сущностей. На одном таком уровне скорость получения ресурсов почти не меняется, независимо от роста других характеристик. Важно следить, чтобы скорость получения ресурсов не превышала спрос на них.

ОТКРЫТАЯ ЭКОНОМИКА подразумевает возможность обмена ресурсами между игроками. Отсюда следует главная ее проблема: появляются слишком бедные и слишком богатые игроки, а ваша задача как разработчика – по возможности уравнивать их, чтобы люди испытывали более или менее одинаковый дефицит ресурсов.

Главная проблема открытой экономики – это сложность баланса. Если ресурсы можно конвертировать, например, в боевую мощь, вместе с неравным распределением ресурсов вы получите и игроков, имеющих преимущества в поединках, в крафте и других элементах игры. А это может уже сломать геймплей.

Чтобы сохранить баланс, помимо создания нехватки ресурсов, есть вариант взять ситуацию под свой контроль, вводя пошлины, налоги, то есть какую-то плату за обменные операции. Другой хороший принцип, например для ММО, – когда персонажам высокого уровня по-прежнему нужны низкоуровневые ресурсы. Тогда обмен ресурсами между разными категориями игроков будет более сбалансированным.

Отдельный инструмент открытой экономики – аукцион. Если игроки могут влиять на стоимость ресурсов, они сами балансируют экономику, выставя адекватные, на их взгляд, цены.

У экономики каждого конкретного игрового ресурса может быть три крайности: профицит (когда у игрока ресурс в изобилии), дефицит (когда его всегда не хватает) или баланс (когда ввод и

вывод игровых ресурсов примерно равны). В одной игре могут по-разному сочетаться все эти виды. Плюс не стоит забывать о навыках игроков: обычно кто лучше играет, у того и ресурсов больше.

ЭКОНОМИКА ПО ПРОГРЕССИИ нужна для любой игры, где персонаж проходит игровой путь, двигаясь по какой-то шкале. Для одиночных игр необходимо рассчитать, как персонаж будет получать деньги, очки опыта и пр. Даже для головоломок – игр, казалось бы, совсем не про экономику – нужно продумать, как часто и при каких условиях игрок будет получать подсказки.

ЭКОНОМИКА ДЛЯ МОНЕТИЗАЦИИ ИГРОКОВ составляется таким образом, чтобы стимулировать игроков платить. Одна из основных задач, особенно для f2p игр, – расчет ARPPU (от англ. average revenue per paying user – «средняя выручка с одного платящего пользователя»), то есть какой доход с игрока должна иметь наша игра, чтобы быть успешной. Исходя из этого и считаются все остальные цифры; подгоняем ли мы экономику через дефицит ресурсов или через возможность экономии времени – вопрос второго плана.

ЭКОНОМИКА В КОНТЕКСТЕ ВЗАИМОДЕЙСТВИЯ ИГРОКОВ – это комплексная вещь. Сюда относятся баланс открытой экономики, логика обмена ресурсами между игроками, аукционы и т. д.

ИГРОВАЯ ЭКОНОМИКА КАК ОСНОВА ГЕЙМПЛЕЯ встречается прежде всего во всевозможных градостроительных симуляторах. В рогаликах^[66] и симуляторах выживания острый дефицит ресурсов тоже определяет игровой процесс, влияя на решения и опыт, а также позволяя ощутить неприветливость игрового мира.

Чаще всего гейм-дизайнеру предстоит описать четыре состояния экономики: старт, основной геймплей, хай-энд и монетизация. **НА СТАРТЕ** игрок только помещен в игровой мир, и необходимо определить, какие ресурсы и возможности у него есть в самом начале пути, чтобы он мог плавно втянуться в игровой процесс. Если это survival, он сразу ощущает нехватку ресурсов; экономика MMORPG, напротив, чаще никак не ограничивает игрока, позволяя

ему в свое удовольствие проходить квесты и знакомиться с игровым миром.

Если в вашей игре экономика является важным элементом **ОСНОВНОГО ГЕЙМПЛЕЯ**, необходимо составить дизайн для каждой шкалы: прогрессия опыта, внутриигровых денег, строительства и т. п. Есть элементы, характерные для конкретного жанра (например, экономических стратегий), им следует уделить особое внимание при проектировании геймплея.

ХАЙ-ЭНД – это этап, когда у игрока уже завершен базовый путь прохождения, который вы заложили. Скажем, в том же survival игрок уже прокачался, построил себе крепкий дом, завел животных, обеспечив пропитание и пр.; или в MMORPG – все возможные боссы побеждены, все желаемые доспехи собраны, а значит, внутриигровые деньги больше не нужны. Хотя в онлайн-играх фаза насыщения обычно сильно отложена из-за долгой прогрессии, связанной с действиями других игроков, необходимо продумать, что же будет удерживать игрока на данном этапе. Если вовремя этого не сделать, то экономика просто схлопывается (пропадают цели и желание играть дальше); при закрытой экономике – валюта обесценивается для одного игрока либо же вообще для всего сервера.

МОНЕТИЗАЦИЯ часто связывает все эти части. Если у игрока ничего не забирать, скоро ему хватит ресурсов на все, и игра потеряет смысл. Если есть проблемы с монетизацией, игра лишится смысла куда быстрее: игроки просто не дойдут до поздних стадий. Для многих MMORPG во время прокачки игроку почти не нужно донатить ^[67] в игру, а после прокачки до топового уровня возможностей для трат становится все больше. Дело здесь в психологии: чем больше люди тратят времени и сил на игру, тем реальнее становятся игровые достижения, тем сильнее привычка и тем сложнее игру оставить.

КАК СЧИТАТЬ ИГРОВУЮ ЭКОНОМИКУ

Есть игры (например, ферма Township), для которых игровая экономика построена на основе баланса ввода и вывода ресурсов, тогда ваша задача – придумать, как сделать игровой процесс интересным. Но если ваша игра монетизируется за счет продажи игровых сущностей, то есть за счет нехватки ресурсов, успех всего проекта будет зависеть именно от корректных расчетов. В этом случае именно желаемые бизнес-показатели станут определять баланс.

Прикидывать, как будет построена игровая экономика, следует на самом начальном этапе разработки, когда еще может не быть даже прототипа. Важно обозначить хотя бы самые базовые вещи: плейтайм (на сколько времени рассчитаны все активности в нашей игре) и ожидаемую прибыль, которую принесут игроки, – особенно это критично для f2p-игр. Некоторые проекты могут позволить себе подождать, пока игроки освоятся, и лишь потом начать зарабатывать; для других, особенно мобильных, критично начать получать прибыль как можно скорее. Для ряда жанров стоит заранее предположить, как игроки психологически воспримут игровую экономику.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1																		
2				Constants		Weapon_type	Bonus	Close C	Ranged	High R	Indoor							
3			Basic HP	200		Pistol	0%	25%	100%	0%	25%					Minimum DPS		19
4			Time to kill, sec	10		Shotgun	0%	75%	25%	0%	75%					Maximum DPS		39
5			Power Bonus	10%		Automatic	10%	50%	100%	0%	25%					DPS rate		2,052632
6			Kills per weapon	3		Melee	50%	50%	0%	0%	100%					Minimum Power		3
7			AoE dmg Debuff	5%		Rifle	50%	0%	50%	100%	0%					Maximum Power		29
8																Power rate		9,378281
9																		
10			1 Ruger LCP	weapon_pistol_ruger	Pistol	0	20	15	0,6	6	1	0	14,0	2	0,8	40	200	3
11			2 Glock G19	weapon_pistol_glock	Pistol	0	22	15	0,6	15	1,2	0	14,0	2	0,8	44	220	3
12			3 Beretta 93R	weapon_pistol_beretta	Pistol	0	24	16	0,6	15	1	0	14,0	2	0,8	45	240	4
13			4 Colt Python	weapon_pistol_python	Pistol	0	26	24	0,7	6	1,4	0	14,0	2	0,8	33	260	4
14			1 Cutoff	weapon_shotgun_cutoff	Shotgun	1	19	33	1	2	1,5	0	12,0	10	0,7	18	200	13
15			2 Remington 870 Express	weapon_shotgun_remington870	Shotgun	1	20,9	29	1	4	1,5	0	12,0	10	0,7	23	220	15
16			3 Mossberg 500 Mariner	weapon_shotgun_mossberg	Shotgun	1	22,8	30	1	6	2	0	12,0	10	0,7	24	240	16
17			4 SPAS-12	weapon_shotgun_spas12	Shotgun	1	24,7	29	0,8	6	2,2	0	12,0	10	0,7	29	260	17
18			1 MP5	weapon_assault_mp5	Automatic	0	22	8	0,3	30	1,5	0	14,0	5	0,3	83	220	9
19			2 M16	weapon_assault_m16	Automatic	0	24,2	9	0,3	20	1,5	0	14,0	5	0,3	81	242	9
20			3 AK-103	weapon_assault_ak103	Automatic	0	26,4	9	0,3	30	1,4	0	14,0	5	0,3	88	264	10
21			4 FN Scar	weapon_assault_scar	Automatic	0	28,6	11	0,3	20	1,5	0	14,0	5	0,3	78	286	11
22			1 Pipe	weapon_melee_pipe	Melee	1	28,5	20	0,7	1	0	0	1,0	45	1,0	45	300	7
23			2 Bat	weapon_melee_bat	Melee	1	31,35	31	1	1	0	0	1,5	90	1,0	32	330	25
24			3 Machete	weapon_melee_machete	Melee	1	34,2	24	0,7	1	0	1	1,0	45	1,0	45	360	9
25			4 Axe	weapon_melee_axe	Melee	1	37,05	37	1	1	0	1	1,5	90	1,0	32	390	29
26			1 Marlin 1894	weapon_rifle_marlin1894	Rifle	0	30	45	1	6	3	0	22,0	5	0,0	20	300	18
27			2 No 10 M24	weapon_rifle_no10m24	Rifle	0	33	53	1	5	3	0	22,0	5	0,0	19	330	20
28			3 Thompson Center Pro Hunter	weapon_rifle_thompson	Rifle	0	36	54	1	6	3	0	22,0	5	0,0	20	360	22
29			4 Dragunov SVD	weapon_rifle_svd	Rifle	0	39	101	2	5	3	0	22,0	5	0,0	12	390	24

Рис. 19. Расчет экономики

Итак, прежде всего мы выписываем все прогрессии по всем ресурсам в нашей игре. После сводим их в единую таблицу, чтобы убедиться, что наши предположения не приводят к несправедливому распределению ресурсов. Важно, чтобы приведенные цифры были четко связаны с геймплеем, это тоже необходимо прописать. Например, мы решили, что игрок получает за убийство крыс 100 золота в час. В этом утверждении есть три пункта: во-первых, что типовой игрок вообще пойдет бить крыс; во-вторых, сколько золота выпадет с одного монстра; и третье – сколько времени игрок потратит на их убийство. Если мы увидим, что игрок получает слишком много ресурса, это нужно исправить. Например, увеличить время на убийство или научить монстра разрушать броню, которую потом придется чинить за ресурсы.

Обычно создается математическая модель расчетов на некоего нашего типового игрока. Предполагаем, что он должен убивать такого-то монстра за 30 секунд, раз в два уровня покупать новый меч и т. д. Отдельно можно сделать расчеты также для казуального и, наоборот, хардкорного игрока, подкручивая цифры под конкретный стиль игры, чтобы хардкорные игроки не слишком быстро забирали

контент, а казуальные не слишком страдали. После запуска игры эти расчеты необходимо сверить с уже реальной статистикой.

Разные виды валюты считают несколькими способами. Во-первых, соотносят получаемые ресурсы со временем: скажем, сколько золота игрок может заработать в час или в минуту. Во-вторых, гейм-дизайнеры сознательно вводят ограничения – потолок, или кап (от англ. cap), количества ресурсов/попыток получить эти ресурсы, больше которых игрок получить не может. Например, ежедневные квесты могут давать хорошую награду, но доступны только раз в сутки. Или после выполнения некоторого количества заданий награда за каждое последующее снижается.

Далее – планка сложности, то есть какого уровня должны быть экипировка или навык игрока, чтобы получать ресурсы выбранным способом. Можно, допустим, пойти на поле и спокойно копать ресурсы, а можно отправиться убивать босса и при этом оказаться в минусе, так как придется тратиться на ремонт доспехов и новые зелья. И последнее – как источник ресурса будет меняться со временем, от действий игрока или разработчиков. Некоторые игры используют обесценивание ряда ресурсов. Другие делают ресурсы исчерпаемыми. На этом, к примеру, основана экономическая часть баланса на картах такой популярной некогда стратегии, как Warcraft.

Все эти вещи имеет смысл выписать в отдельную таблицу, так как она помогает достичь понимания, какие ресурсы в игре могут использоваться игроками несправедливо. Обычно, если ресурс ценен, игроки выбирают тот способ добывать его, при котором они гарантированно не проиграют и получают максимальную прибыль. Такой подход верен для игроков, подходящих к проблеме дефицита ресурсов рационально, и для MMORPG или survival-игр это часто так. У казуальных игр может быть аудитория, которая вместо практичного оружия всегда выбирает прическу персонажа, потому что она красивая, из-за чего их всех убивают, и им становится тяжело и скучно играть. Здесь игроки менее рациональны, и предсказать, как им захочется добывать и тратить ресурсы, сложнее.

Поэтому простые проекты редко дают много играть в экономику, чтобы не раздражать казуальную аудиторию.

В итоге есть два противоположных подхода к своим игрокам: или «игрок должен страдать», или «мы стараемся потакать игроку во всем». Для ПК-игр в большинстве своем верен первый подход, а вот мобильные казуальные игры, яростно сражающиеся за удержание аудитории, пытаются всячески игрокам угодить и не напрягать их. Хотя во free-to-play играх довольно часто встречаются paywall^[68] или энергия (необходимость заплатить, чтобы поиграть еще прямо сейчас, а не ждать). Конечно, это верно не всегда: есть и мобильные игры с жесткой экономикой, и суперлояльные игры на ПК.

Считается, что сложная игровая экономика – это что-то, что игрок должен преодолеть, потратив время и силы на осознание, по каким правилам она работает, чтобы играть хорошо и получать удовольствие. В отличие от монетизации pay-to-win^[69], в большинстве своем игроки нормально относятся к сложной игровой экономике, если она не подразумевает много гринда. Такие игры могут отпугивать казуальных игроков, но в целом воспринимаются хорошо.

Бывает, что разработчики пытаются предугадать все действия игрока. Это почти невозможно, и строить на таких предположениях баланс – плохая идея. Эффективнее сосредоточиться на ограничениях, так вы будете уверены, чего пользователь хотя бы точно делать не будет. Яркий пример ограничений можно увидеть в батлерах. Часто их разработчики никак не ограничивают активности игрока, но отлично просчитывают, как далеко может зайти игрок каждый день. Эта механика хорошо реализована в *AFK Arena*.

Каждый день игрок может проходить сколько угодно уровней башни, продвигаться по миссиям так далеко, как только сможет. Может сражаться на арене, покупать героев – делать практически что угодно. Хотя есть и жестко ограниченные части геймплея, такие как мистический лабиринт – три уровня путешествия, где герои не восстанавливают здоровье после битвы. Пройдя его полностью,

ждите два дня до обновления. В большинство остальных активностей можно играть бесконечно. И в начале игры кажется, что так оно и будет. Но вот герои оказываются уже недостаточно сильны для прохождения следующих уровней, а на прокачку ресурсов не хватает, и нужно ждать следующей ежедневной награды. И дейли-квесты засчитываются лишь за первое прохождение той или иной активности в день. Все остальные уже не принесут дополнительных наград. И вроде бы получается, играй сколько хочешь, но на практике игра оставляет лишь 10–15 минут геймплея в день. Такие ограничения позволяют разработчикам подключать монетизационные механизмы.

При работе над игровыми механиками без плейтестов не обойтись. Пара советов: во-первых, меняйте за раз что-то одно, чтобы всегда понимать, какие решения имеют последствия, ведь в игре элементы зачастую взаимосвязаны. Второй важный момент: изменения должны быть достаточно заметными, чтобы статистика сразу давала понять, в нужном ли направлении мы движемся; потом уже можно подправить значения в меньшую сторону для итогового баланса. Excel – доступное и удобное средство для работы над балансом и экономикой, он позволяет использовать динамические таблицы, генерировать случайные числа, учитывать состояния сотен классов и юнитов. Вам нужно освоить и полюбить эту программу.

Мир знает примеры интересных и успешных разбалансированных игр, так что важно в погоне за стремлением уравновесить все игровые сущности, не потерять основные цели и ощущения от самой игры.

Производство контента

КОНФИГУРИРОВАНИЕ КОНТЕНТА

Производство контента – неотъемлемая часть работы геймдизайнеров. В данном случае под контентом мы понимаем различные приобретаемые игроками сущности, вроде новых юнитов, экипировки, а также расходные ресурсы (зелья, гранаты и т. д.) Допустим, у нас уже есть ГДД, в котором подробно описаны игровые механики и логика производства контента. Допустим, у нас даже посчитан баланс и все цифры, необходимые для имплементации (практической реализации) этого контента в игру. Как же заставить программу, не имеющую образного мышления, понять и воспроизвести все написанное, особенно если гейм-дизайнер не умеет писать код?

К счастью, навыки программирования для этого не потребуются. Более того, создание контента программистами считается плохой практикой и называется «хардкодом». Под этим словом мы понимаем ситуацию, когда программист добавляет динамические элементы в код: цифры, объекты, описания и т. д. Во многих случаях в коде допустимо хранить формулы, но продвинутые гейм-дизайнеры производят все расчеты в Excel и отдают уже посчитанные табличные данные. Разумеется, если это допустимо в конкретной игре. Очень удобно посчитать в Excel характеристики всех построек экономической стратегии, но довольно трудно убрать из кода формулы работы заклинаний в MMORPG. Хотя стоит отметить, что это вполне возможно и в идеальной ситуации так и нужно сделать. И, скорее всего, невозможно отказаться от формул при расчете динамики в таких играх, как, например, *Overwatch*.

Как вы уже, наверное, поняли, речь в этой главе пойдет про **КОНФИГИ**. Это структурированный набор данных, описывающий

любые внутриигровые сущности на понятном для выбранной программы языке. Конфигурационные файлы нельзя назвать программным кодом. Скорее это описание, созданное с помощью стандартизированных языков. К таким языкам относятся, в частности, языки разметки текста. К примеру, XML.

```
1  <!-- Создано с помощью MIIP_Gamedesign_Craft_calculator_v.1.xlsx -->
2  <data>
3    <recipe type="recipe_molotov_coctail"
4      outcome="molotov_coctail"
5      quantity="1"
6      time="1"
7      rarity="Common">
8      <consume>
9        <ingredient item="bottle" amount="1" />
10       <ingredient item="dirty_rags" amount="1" />
11       <ingredient item="Fuel" amount="3" />
12     </consume>
13   </recipe>
14   <recipe type="recipe_crafted_gun"
15     outcome="crafted_gun"
16     quantity="1"
17     time="2"
18     rarity="Legendary">
19     <consume>
20       <ingredient item="weapon parts" amount="2" />
21       <ingredient item="instruments" amount="3" />
22     </consume>
23   </recipe>
24 </data>
25
```

Рис. 20. Визуальный пример XML конфига

Если в игре лучница должна пройти игровой уровень размером в 1 экран телефона (допустим, поле 12*20 условных клеточек), преодолевая препятствия и убивая врагов, в конфиге этого уровня будет написано примерно следующее: на клеточке [2;2] и [5;6] стоят объекты номер 4 (кубики), на [7;8] – забор (тоже объект с присвоенным ему номером или названием). Игра прочитает эти координаты и воспроизведет препятствия, информацию о которых

сможет найти по ссылке на файл «level_1». На этапе загрузки она также «подтянет» файл со списком всех объектов уровня (а может, и всех уровней вообще), где указано, что, например, камень – это препятствие, размером 1×1, «непроходимый», ссылка на изображение такая-то... Для каждой сущности нужно создать такое простое описание. Для мобов указать важные параметры: сколько у него здоровья, количество урона, тип атаки, как он выглядит, с какого расстояния атакует и т. д.

Загрузка – это как раз время, необходимое программе, чтобы в том числе «прочитать» файлы нужного уровня и воспроизвести его с помощью игрового движка в памяти устройства. Самой игры, управления и графических ассетов в конфигах нет. С переходом на следующий уровень игра опять будет обращаться к файлам, чтобы загрузить конфигурацию новой локации и другие параметры.

В зависимости от средств производства и хранения контента можно пользоваться разными техническими инструментами. Мы приведем наиболее распространенные.

XML

Одним из самых простых и популярных инструментов для описания неких структур считается XML (от англ. eXtensible Markup Language). Как мы уже говорили, это язык разметки, то есть, в отличие от кода (языка программирования), мы не отдаем с его помощью никаких команд. Он близок к HTML^[70], который используют для создания веб-сайтов и интерфейсов: там тоже нет инструкций как таковых; HTML – описательный язык, позволяющий рассказать, как внешне должен выглядеть сайт.

Давайте рассмотрим пример. В нашей «Очень счастливой ферме» есть 100 видов ресурсов, 40 различных животных, 75 строений и т. д. Все эти игровые сущности имеют собственные параметры, собираемые из разных компонентов, и должны как-то

конфигурироваться и храниться. У нас есть здание «Загон для барашка». Оно обладает параметрами: размер, доступные животные, визуальный арт и т. д. Для его постройки необходимо 2 железа, 100 монет и 3 дерева. Для дальнейшего улучшения – еще какие-то ресурсы. Мы можем описать это в виде списка с соответствиями: уровень здания – список ресурсов.

```
<building name="zagon_baran">  
  <size width="2" height="3"/>  
  <upgrade level="1">  
    <resource type="iron">2</resource>  
    <resource type="lumber">3</resource>  
  <money>100</money>  
  </upgrade>  
</building>
```

Теперь программа сможет прочитать это описание и отобразить информацию о строительстве данной сущности в интерфейс пользователя. Или же наоборот: если игрок нажимает «Создать загон для барашка», обратится к этим файлам для проверки, выполнены ли все условия, прежде чем создавать строение.

```

1 <generate_items_pool_m type="">
2   <pool>
3     <item type="card_mercenary_fair_soldier_1_count_item" weight="62" category="basic" />
4     <item type="card_mercenary_fair_gunner_1_count_item" weight="62" category="basic" />
5     <item type="card_mercenary_fair_tank_1_count_item" weight="62" category="basic" />
6     <item type="card_mercenary_fair_sniper_1_count_item" weight="62" category="basic" />
7     <item type="card_mercenary_fair_thrower_1_count_item" weight="62" category="basic" />
8     <item type="card_mercenary_fair_jeep_1_count_item" weight="62" category="basic" />
9     <item type="card_base_artillery_count_item" weight="62" category="basic" />
10    <item type="card_reputation_light_infantry_health_buff_1_buff_item" weight="62" category="basic" />
11    <item type="card_reputation_light_infantry_damage_buff_1_buff_item" weight="62" category="basic" />
12    <item type="card_reputation_heavy_infantry_health_buff_1_buff_item" weight="62" category="basic" />
13    <item type="card_reputation_light_infantry_ability_damage_buff_1_buff_item" weight="62" category="basic" />
14    <item type="card_reputation_heavy_infantry_damage_buff_1_buff_item" weight="62" category="basic" />
15    <item type="card_reputation_all_infantry_health_buff_1_buff_item" weight="62" category="basic" />
16    <item type="card_reputation_heavy_infantry_ability_damage_buff_1_buff_item" weight="62" category="basic" />
17    <item type="card_reputation_all_infantry_heal_buff_1_buff_item" weight="62" category="basic" />
18    <item type="card_reputation_all_infantry_damage_buff_1_buff_item" weight="62" category="basic" />
19    <item type="card_reputation_all_infantry_ability_radius_buff_1_buff_item" weight="62" category="basic" />
20    <item type="card_reputation_light_infantry_health_buff_2_buff_item" weight="62" category="basic" />
21    <item type="card_reputation_light_infantry_damage_buff_2_buff_item" weight="62" category="basic" />
22    <item type="card_reputation_heavy_infantry_health_buff_2_buff_item" weight="62" category="basic" />
23    <item type="card_reputation_heavy_infantry_damage_buff_2_buff_item" weight="62" category="basic" />
24    <item type="artefact_constant_heavy_armor_damage_1_count_item" weight="62" category="basic" />
25    <item type="artefact_constant_heavy_infantry_damage_1_count_item" weight="62" category="basic" />
26    <item type="artefact_constant_heavy_armor_health_1_count_item" weight="62" category="basic" />
27    <item type="artefact_constant_light_infantry_damage_1_count_item" weight="62" category="basic" />
28    <item type="artefact_constant_light_armor_damage_1_count_item" weight="62" category="basic" />
29    <item type="artefact_constant_production_money_1_count_item" weight="62" category="basic" />
30    <item type="artefact_constant_heavy_infantry_health_1_count_item" weight="62" category="basic" />
31    <item type="artefact_constant_light_infantry_health_1_count_item" weight="62" category="basic" />
32    <item type="artefact_constant_light_armor_health_1_count_item" weight="62" category="basic" />
33    <item type="artefact_constant_production_food_1_count_item" weight="62" category="basic" />
34    <item type="artefact_constant_production_fuel_1_count_item" weight="62" category="basic" />
35    <item type="light_mercenary_machinery_exchange_intel_collection_count_item" weight="62" category="basic" />
36    <item type="heavy_mercenary_machinery_exchange_intel_collection_count_item" weight="62" category="basic" />
37    <item type="support_elite_exchange_intel_collection_count_item" weight="62" category="basic" />
38    <item type="resource_box_exchange_intel_collection_count_item" weight="62" category="basic" />
39    <item type="diamond_exchange_intel_collection_count_item" weight="62" category="basic" />
40    <item type="light_infantry_exchange_intel_collection_count_item" weight="62" category="basic" />
41    <item type="heavy_infantry_exchange_intel_collection_count_item" weight="62" category="basic" />
42    <item type="light_machinery_exchange_intel_collection_count_item" weight="62" category="basic" />
43    <item type="heavy_machinery_exchange_intel_collection_count_item" weight="62" category="basic" />
44    <item type="support_exchange_intel_collection_count_item" weight="62" category="basic" />
45    <item type="soldier_collectable_intel_collection_count_item" weight="62" category="basic" />
46    <item type="gunner_collectable_intel_collection_count_item" weight="62" category="basic" />
47    <item type="base_tank_collectable_intel_collection_count_item" weight="62" category="basic" />

```

Рис. 21. Фрагмент XML-кода, описывающего содержимое лутбокса браузерной игры

XML несложен в написании. Есть специальные программы, например, Notepad++ или JetBrains WebStorm, позволяющие сделать данные нагляднее, используя отступы, разные цвета и т. д. Можно создавать файлы в редакторах и отдавать в игровой движок, а можно писать сразу в редакторе, который вы используете для работы с Unity (к примеру, MonoDevelop).

БАЗЫ ДАННЫХ И SQL

Базы данных позволяют создавать не просто текстовые файлы. В отличие от XML, это уже особым образом структурированные данные, работа с которыми происходит через движок базы данных (БД), позволяющий к этим данным обращаться. У каждого движка свой язык, на котором он принимает команды. Самый популярный тип баз данных в игровой индустрии – SQL (от англ. Structured Query Language – «язык структурированных запросов»).

База данных – это набор взаимосвязанных таблиц. В визуальном редакторе создаются таблицы, куда вносятся данные. В SQL это строка с заголовками, значениями и т. д.

Бывают разные вариации этого языка, используемые разными движками СУБД (MySQL, PostgreSQL, MsSQL, SQLite и другие), но логика одна и та же. Основные команды для работы с БД можно пересчитать по пальцам:

SELECT – выгрузить из базы нужные данные;

INSERT – вставить в базу новую запись;

REPLACE – заменить имеющуюся запись;

DELETE – удалить вхождения данных из базы.

Конечно, в работе вы столкнетесь с гораздо большим набором команд для транзакций^[71], но при помощи этих можно закрыть 90 % простых задач. Для более сложных запросов – к примеру, хранимых процедур – потребуется либо более глубоко изучить SQL, либо обратиться к вашему аналитику, если такой есть в команде. Аналитик обязательно знает множество команд и нюансов работы с БД.

Помимо SQL, существуют и другие варианты логики хранения и обработки информации в БД. Мы не будем их касаться, потому что они более редкие и сложные в освоении, – лучше обратиться к специализированной литературе.

Для хранения большого объема данных, особенно если предполагается, что эти данные постоянно меняются, лучше выбирать базу данных, а не XML. Допустим, в нашей игре миллион игроков, и по каждому необходимо собирать метрики. Хранить динамические данные в XML-файле не так эффективно; база данных

позволяет быстрее оперировать большим объемом динамической информации. Но это не означает, что статическая информация также должна быть размещена в БД.

Есть примеры сочетания обоих способов: база для динамических данных и XML для статических. Например, гейм-дизайнер решил, что в игре должно быть 20 уровней, и описал их в XML. Эти данные меняться не будут, можно оставить информацию и в этом формате. А вот если нас интересует, на каком уровне сейчас находится конкретный игрок, сколько у него внутриигровой валюты и другие переменные, нужно обратиться к базам данных.

XML легко писать, но тяжело редактировать; базы данных удобнее и быстрее в использовании, но не все гейм-дизайнеры умеют с ними работать. Для онлайн-проектов обычно используются базы данных. Старые игры часто пользовались XML, так как не надо было ничего подгружать в реальном времени.

Базы данных заточены под быструю передачу информации, XML – под удобство описания. Существуют и другие языки, кроме SQL и XML: JavaScript, ActionScript; в Unreal Engine для описания сценариев используют блупринты. Выбор инструментов весьма широкий и зачастую определяется особенностями игрового проекта и личными предпочтениями.

АВТОМАТИЗАЦИЯ РАБОТЫ

Итак, описание XML кода и структуру базы обычно делают программисты или (в крупных студиях) архитекторы баз данных на основе документов от гейм-дизайнера. После того как вся экономика посчитана, для чего чаще всего используется Excel, гейм-дизайнеру предстоит внести в игру получившиеся значения.

Можно, конечно, заниматься «мартышкиным трудом», переписывая каждую цифру вручную. Многие разработчики так делали, пока не допустили первые ошибки. Именно вероятность

допустить ошибку, просмотреть что-то и здоровая лень зародили мысль об автоматизации.

Гейм-дизайнеру очень важно уметь создавать собственный удобный инструментарий, чтобы упростить свою работу. Гораздо эффективнее было бы потратить время и написать программу для автоматизации выгрузки данных из Excel. Помимо ускорения работы, мы можем проработать проверку важных параметров на валидность. Скажем, цена морковки (либо другого ресурса) в нашей ферме не может быть меньше 0. Если такая ситуация произошла, нужно, чтобы Excel или программа выгрузки обратили на это внимание (например, подсветив неправильные значения красным). В некоторых компаниях такие помогающие гейм-дизайнеру программы пишут программисты.

Если гейм-дизайнеру не хватает навыков работы с Excel, сделать это самостоятельно, особенно при работе с базами данных, может быть проблематично. Фактически ему не удастся оградить себя от ненужного труда, ошибок и лишних вопросов со стороны коллег. Руководство студий часто просит работать больше, и задача гейм-дизайнера – вовремя самому задуматься над тем, как оптимизировать свою работу. Реализовав эти идеи, можно раз и навсегда переложить обязанности по проверке данных, расчетам, формированию XML-кода и т. д. на технические инструменты, тем самым автоматизировав часть рабочих процессов.

На скриншоте приведен пример данных из экономической стратегии. Это пример скрипта на языке VBA, который не только проводит расчеты, заполняя недостающие ячейки в Excel, но и создает SQL, XML или JSON код для экспорта в игру. Простой VBA-скрипт (упрощенная реализация языка программирования Visual Basic) пробегается по этому списку и генерирует текст SQL-скрипта, который можно просто скопировать в SQL-редактор и выполнить. Он сам переписет нужную часть информации в соответствии с тем, что вы прописали в Excel.

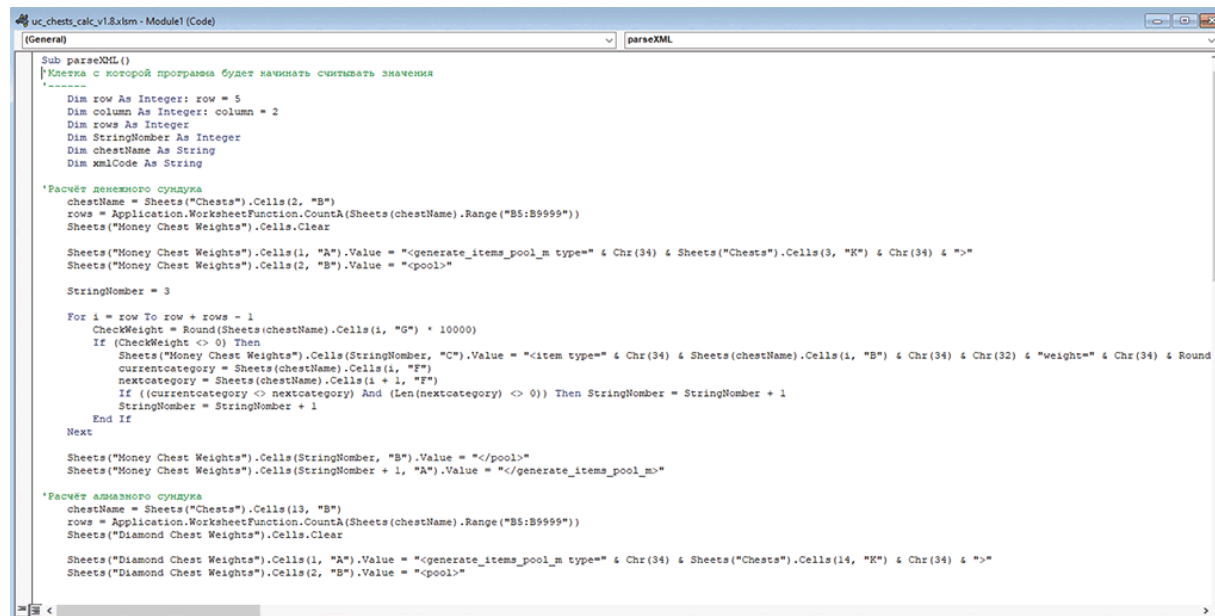


Рис. 22. Пример данных из экономической стратегии

Для многих ГД освоение встроенного языка VBA является относительно трудоемкой задачей. Мы рекомендуем однажды найти в себе силы и изучить его, навсегда упростив себе жизнь. Однако, если вам этого категорически не хочется, Excel позволяет решать подобные задачи и более простым путем, например через обычные формулы:

=”INSERT into `buildings` (” & B1 & ”,” & B2 & ”);”

Надеюсь, нам удалось донести до вас важность создания собственных инструментов. Эта логика верна для любых задач: прежде чем что-то делать, задумайтесь: может быть, вы можете автоматизировать процесс, снизив вероятность ошибки, сохранив свое время и в итоге спасая собственные нервы от перегрузки. Все это – часть общей культуры производства.

АРТ

Любой гейм-дизайнер, даже не планирующий самостоятельно работать над графикой, должен понимать нюансы процессов разработки игрового арта. Чтобы верно рассчитывать время создания графического контента, а также грамотно составлять ТЗ, необходимо определить, какие виды арта существуют и в чем их особенности. В крупных компаниях для создания каждого типа арта существует свой человек или отдел, в маленьких командах эти должности часто совмещаются.

Обычно художники лучше гейм-дизайнеров знают, что и как рисовать. Поэтому нет смысла перечислять, как должен выглядеть каждый камень или стол в вашей игре. Очень редко возникает необходимость уточнить, как именно персонаж должен держать меч или забор какого цвета должен быть в создаваемой игровой деревушке. Назначая задание на арт, важно объяснить, какие ассоциации и ощущения должна вызывать картинка, желательно с примерами, а также обозначить, с какими элементами арт должен сосуществовать.

КОНЦЕПТ-АРТ

Обычно концепт-арт создается первым и не имеет самостоятельной ценности. Это часть комплексной задачи по созданию визуального стиля, а не финальная картинка. Концепт-художник должен уметь быстро делать наброски и предлагать разные варианты исполнения того или иного игрового элемента, чтобы гейм-дизайнер и другие члены команды могли от чего-то оттолкнуться при поиске оптимального решения. Такие рисунки могут выполняться без использования специальных программ, без цвета или деталей. Техника выполнения концепт-арта может быть

самой разной: можно рисовать от руки, а можно использовать фотографии или фрагменты чужих работ, чтобы сэкономить время. Работа концепт-художника обычно остается внутри компании: игроки видят ее только в том случае, если этот арт используется для продвижения.

ДИЗАЙН ИНТЕРФЕЙСОВ

Дизайн интерфейсов (UI/UX) направлен на игрока. Сюда можно отнести все, что игрок видит на экране вне рамок игрового процесса: меню, кнопки, иконки и пр. Именно он помогает легко разобраться с управлением и окружением, правильно расставляет акценты, поэтому имеет свои правила и процессы. Речь идет не только о внешнем виде элементов интерфейса, но и о логике и опыте взаимодействия с ними игрока. Для такой работы критически важно иметь общую стилистику интерфейсов игры. В средних и крупных компаниях дизайном интерфейсов обычно занимаются отдельные специалисты.

Прежде всего гейм-дизайнером составляется задание с полным описанием фичи и ее логики: что это, какие предполагаются виды взаимодействия с игроком, как это должно выглядеть, возможности перехода и т. д. На следующем этапе хорошей практикой считается сделать макет интерфейса без финального оформления. Сначала это просто система переходов, где подробно прописано, что видит игрок, какие методы используются, чтобы облегчить понимание игроком, какой следующий шаг ему необходимо сделать, какие средства помощи игроку предусмотрены, если он не понимает, что происходит. Для этого используют специальные программные средства, например Figma^[72], которые позволяют собирать такие схемы переходов с анимациями. Это дает гейм-дизайнеру возможность проверить, все ли работает так, как он задумал, или нужно внести какие-то изменения.

Вообще, гейм-дизайнер и человек, отвечающий за игровой интерфейс, должны много общаться, чтобы иметь одинаковое представление о том, какие вещи играют ключевую роль и должны быть выделены, как та или иная фишка выглядит с точки зрения логики и как игроки должны ее использовать. Полезными документами здесь окажутся юзер-кейс или юзер-стори из ГДД, где гейм-дизайнер предполагает, какие шаги может предпринять игрок, что он должен увидеть в тот или иной момент игры.

Не все компании располагают ресурсами для проведения профессионального UX-тестирования в лаборатории, с отслеживанием пульса, движений глаз пользователя и пр. Но чтобы оценить, насколько понятным и приятным получается интерфейс, необходимо собирать отзывы. Если пользователь действует не так, как задумал гейм-дизайнер, значит, что-то уже пошло не так и принятые решения требуют пересмотра.

Хороший интерфейс должен быть незаметен для игрока. Нужно помнить, что есть общие для всех привычки, например закрытие окна – красный крестик в правом верхнем углу. Чтобы поменять его цвет или расположение, нужны веские основания, так как пользователю будет некомфортно привыкать к новой конфигурации. Зачастую имеет смысл делать именно привычные игрокам иконки, окошки, крестики и т. п. *World of Warcraft* приучил игроков к тому, что за восклицательным знаком игрока ждет квест, и эту практику переняли почти все игровые жанры. Если игроки интуитивно понимают, какая полоска на экране отвечает за здоровье, а какая – за очки опыта, в 90 % случаев нет необходимости придумывать что-то новое.

Важно не перегружать интерфейс, чтобы не вызывать ощущения отчужденности. Люди хотят играть, а не нажимать бесконечные кнопки в меню. Не всегда художники знают все нюансы, поэтому, составляя ТЗ, гейм-дизайнер должен обозначить, на что следует обратить внимание, работая над тем или иным элементом.

Нередко интерфейс сопровождается системой обучения, встроенной в игровой процесс. Помимо игрового туториала, пользователю нужно понимание, в какой раздел меню он может заходить, чтобы совершить то или иное действие. С помощью подсказок игрок выполняет те же действия, как если бы он полноценно играл, но задания он получает с разъяснениями и в определенном порядке. В мобильных стратегиях, например, часто подсвечивают элементы, информируя: чтобы построить здание, игроку нужно нажать на него.

Из старых решений, которые сейчас почти не встретишь, – кнопка помощи F1, открывающая окно с подсказками и правилами. Ее почти не используют, так как от игрока требуется не только лишнее действие, но еще и признание, что он не знает, что делать. Под видом мягкой F1 часто встречается не вызывающая раздражения система интерактивных подсказок. Например, если игрок подошел к двери, но почему-то не может ее открыть, сразу или через заданное время появится подсказка: «Нажмите E, чтобы открыть». Вы наверняка встречали подсказки в повествовании, когда персонаж игрока или другие NPC дают бесценные советы типа: «Наверное, если обрушить эти камни, здесь получится пройти».

Как правило, встречается такое разделение UI/UX.

UX-дизайнер создает саму логику интерфейса, определяет положение, размеры и значение тех или иных элементов. Такой специалист может попутно подбирать и графические решения (степень детализации иконок, контрастность элементов и так далее), так как в данном случае это решения в рамках юзабилити (способности продукта быть понятным, используемым и привлекательным).

UI-дизайнер создает оформление на основании макета, созданного UX-специалистом. UI-художник следит за соответствием стилистики всех интерфейсов игры и логики UX-дизайна, финализирует графику или использует уже готовые элементы из библиотеки интерфейсов игры.

Нередко эти две позиции совмещает один специалист – UX/UI-дизайнер.

ПРОМО-АРТ

Этот арт для всевозможных маркетинговых акций и продвижения игры часто создается отдельно. Баннеры и плакаты, иконки для сторов, обработанные скриншоты и прочее часто создаются отдельными специалистами, близко связанными с маркетинговым отделом. Такой промохудожник, в отличие от концепт-художника, обычно выполняет свою работу с максимальным вниманием к деталям и качеству – в расчете на то, что этот арт будет использован для рекламных кампаний.

ИГРОВОЙ АРТ И ТЗ ДЛЯ ХУДОЖНИКОВ

К игровому арту относятся: персонажи, окружение, игровые локации – то есть то, что пользователь увидит непосредственно внутри самой игры. Чтобы грамотно составить ТЗ для художника, прежде всего нужно четкое понимание, что же мы заказываем (персонаж-спаситель мира, за которого будем играть, спецэффект от вспышки магического взрыва или арт-зелья, который игрок может купить за реальные деньги), ведь подходы могут сильно отличаться.

Сначала необходимо обозначить общее определение арт-стиля. Даже самое подробное описание внешнего вида героя и тех чувств, которые он должен вызывать у игроков, не сможет ответить на фундаментальные вопросы: 2D или 3D, мультяшная, стилизованная или фотореалистичная графика, технические требования (например, 3D-модели должны соответствовать заданному полигонажу, а 2D-изображения – разрешению экрана). Арт-стиль выбирается на самом

раннем этапе производства, в игровых студиях за это решение отвечает арт-директор.

Если выбран фотореализм, нужно помнить о так называемом эффекте «зловещей долины»^[73]. Чем ближе что-то к изображению реального человека, тем более явными выглядят любые несоответствия. Для мультяшного монстра прощательно иметь нереалистичную шерсть или хвост, изгибающийся под странным углом, иногда это даже помогает передать характер и атмосферу. Для реалистичной 3D-модели человека любые несоответствия будут смотреться дико и могут вызывать отвращение.

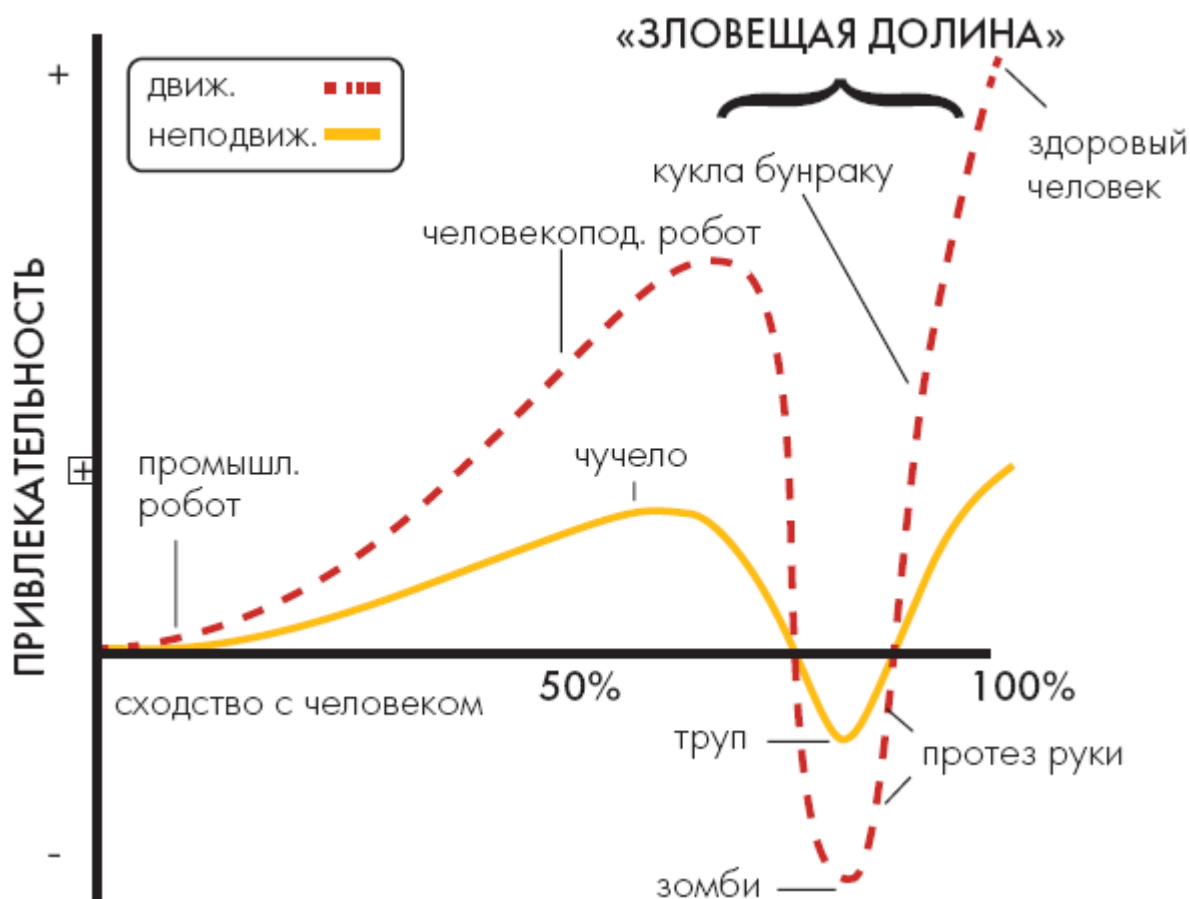


Рис. 23. Зловещая долина

Также необходимо ответить на вопрос, как пользователь будет видеть игровой мир. Вариант от первого лица, когда мы смотрим глазами персонажа, позволяет сэкономить ресурсы на моделировании и анимации главного героя. С другой стороны, так как игрок может подойти к любому предмету в упор, все предметы и окружение должны быть выполнены в соответствующем качестве.

Вид от третьего лица обычно предполагает, что мы смотрим из-за плеча персонажа. Тут уже нельзя обойтись пистолетом в пол-экрана, разработчики должны показать все анимации персонажа. Плюс игрок получает дополнительное понимание пространства, так что требуется большая работа с камерой: ведь, например, монстр может спокойно подойти сзади, и игрок об этом не узнает.

Вариант фиксированной камеры 3/4 сильно экономит усилия по сравнению с камерой свободной. Игрок не приблизит и не выкрутит объект, мы всегда знаем высоту и угол наклона камеры.

РАЗРАБОТКА 3D-АРТА

Давайте разберем **ПАЙПЛАЙН РАЗРАБОТКИ 3D-АРТА ДЛЯ AAA-ПРОЕКТОВ**. Создание объектов и техники здесь обычно требует несколько иных процессов, нежели создание персонажей. AAA-пайплайн подразумевает большой процесс по поиску референсов, созданию и оптимизации модели для экспорта в игровой движок. Технические требования и программы могут меняться со временем, но последовательность этапов создания модели универсальна.

Объекты и технику зачастую основывают на реальных референсах. Если для стратегии или шутера нам понадобится арт реально существующего вертолета, составляется workbook. Это большой документ, где собраны описания нужной нам модели, подходящие фотографии вертолета с разных сторон. Следует заранее обозначить, какие детали необходимо сохранить, а какие допустимо не переносить или переносить с изменениями.

Для игр с более стилизованной или мультяшной графикой потребуется концепт-художник, который сделает наброски нашего вертолета с разных ракурсов в выбранном стиле. После чего заказчик или арт-директор могут выбрать и принять рисунок или же попросить внести те или иные изменения.

На следующем этапе производства эти наработки должны превратиться уже в полноценное ТЗ для художника, который будет создавать модель. Обычно оно включает в себя следующее:

- ссылка на workbook или концепт-арты;
- линейные размеры. Хотя в процессе производства есть возможность масштабировать модели, лучше сразу задавать их размеры, что позволит создавать контент в необходимом качестве;
- технические требования. Намного проще опускать качество графики, чем потом поднимать его, поэтому лучше сразу заложить требования по максимуму. Чтобы оценить размеры и качество моделей, часто делают предварительные наброски всего экрана для понимания, что игрок будет видеть лучше всего. Например, его собственный вертолет, обвешанный красивыми скинами, должен впечатлять; к строениям и деревьям на земле, на которые он смотрит с большого расстояния сверху, не будет таких больших требований. При этом важно и не переоценить возможное качество, иначе время на создание соответствующего контента будет потрачено впустую;
- требования к анимации. Важно описать, как в игре должны функционировать создаваемые модели, как они будут взаимодействовать друг с другом.

Первый этап работы 3D-моделлера называется **BLOCKOUT**. По сути, это набросок, базовая геометрическая форма из примитивов (кубы, цилиндры, сферы) без деталей. На этапе блокинга используют только крупные и средние формы. Основная задача – убедиться, правильно ли все рассчитано с точки зрения базовой геометрии и масштаба, попали ли мы в силуэт, стилистику и пропорции.

Далее мы добавляем к получившейся модели основные детали, это называется детализированный **ДРАФТ**. Так можно проверить, читается ли, что это за модель, насколько выразительной она получается, все ли ключевые составляющие мы добавили, насколько получилось похоже на референс. После дальнейшей детализации переделывать модель будет долго и затратно, лучше исправить все недочеты на этом этапе.

После того как мы приняли драфт, мы переходим к следующему этапу – созданию **HIGH POLY**-модели. Добавляем к модели все обвесы, мельчайшие детали, но пока без текстур. На этом этапе нет ограничений по полигонам. В итоге должна получиться детализированная высокополигональная модель в максимальном качестве. В самой игре High Poly-модель использоваться не будет, ее нужно оптимизировать.

Следующий этап – генерация **LOW POLY**. Это низко детализированная копия получившейся модели, именно ее мы будем экспортировать в игру. Задача на этом этапе – оптимизировать тяжелую и требующую много ресурсов High Poly для того, чтобы можно было свободно работать с ней внутри игрового движка. В играх рендеринг кадра происходит в реальном времени, поэтому на данном этапе мы должны упростить модель, чтобы впоследствии не тратить на это слишком много ресурсов. Каждый элемент Low Poly выполняет свою функцию (например, влияет на силуэт), здесь нет лишних деталей. Процесс создания Low Poly из High Poly называется ретопологией. Именно Low Poly нам предстоит «разворачивать», «печь» и текстурировать.

Иногда процессы создания Low Poly и High Poly могут меняться местами в зависимости от особенностей проекта и создаваемых моделей.

Чтобы текстуры накладывались на модель корректно, нужно создать **UV-РАЗВЕРТКУ**. В детском саду многие из нас склеивали игральные кубики из бумаги, предварительно нарисовав их на

плоскости. Развертка – это обратный процесс, когда мы разбиваем 3D-модель на плоские элементы.

Она нужна, чтобы по ней можно было рисовать текстуры и переносить всю сложную детализацию с High Poly на Low Poly с помощью **«ЗАПЕКАНИЯ» КАРТ НОРМАЛЕЙ**. Запечка позволяет имитировать все красивые неровности, фактуры и другие детали на легкой Low Poly, с помощью бликов создавая иллюзию неровностей.

Далее модель экспортируется в движок, параллельно для нее создаются текстуры. Есть много видов текстур, с помощью которых мы передаем материал, из которого сделан предмет, его цвета, то, как он отражает свет, и т. д. Создание 3D-модели в мультяшной графике в этом плане проще – упор делается на стилизованное изображение, а не на сложную реалистичную передачу света и теней. Здесь часто используются готовые технические решения, исходя из наших потребностей: может быть, мы хотим, чтобы все выглядело как рисунок от руки, или нам нужен арт в стиле нуар, или же нам важны контрастные тени и т. д.

АНИМАЦИЯ

Теперь наш вертолет есть в игре, но работа над ним еще далеко не закончена. Прежде всего стоит задача по анимации движущихся частей: например, у него должен крутиться винт. Как правило, в движках у модели есть список поддерживаемых анимаций. С технической точки зрения, анимация – довольно сложный процесс, поэтому крупные компании нанимают для этого отдельных специалистов-аниматоров. Попробуем вкратце ознакомиться, из чего же он состоит.

С точки зрения гейм-дизайнера, ставящего задачу художникам, важно не только понимать, как должен выглядеть сам объект, но и заранее предполагать, что эта модель умеет делать и как она будет взаимодействовать с другими игровыми сущностями. Именно от

этого будут зависеть необходимые анимации и визуальные эффекты: что к ней крепится, как и при каких обстоятельствах она ломается, с каким звуком начинает работу и прочее.

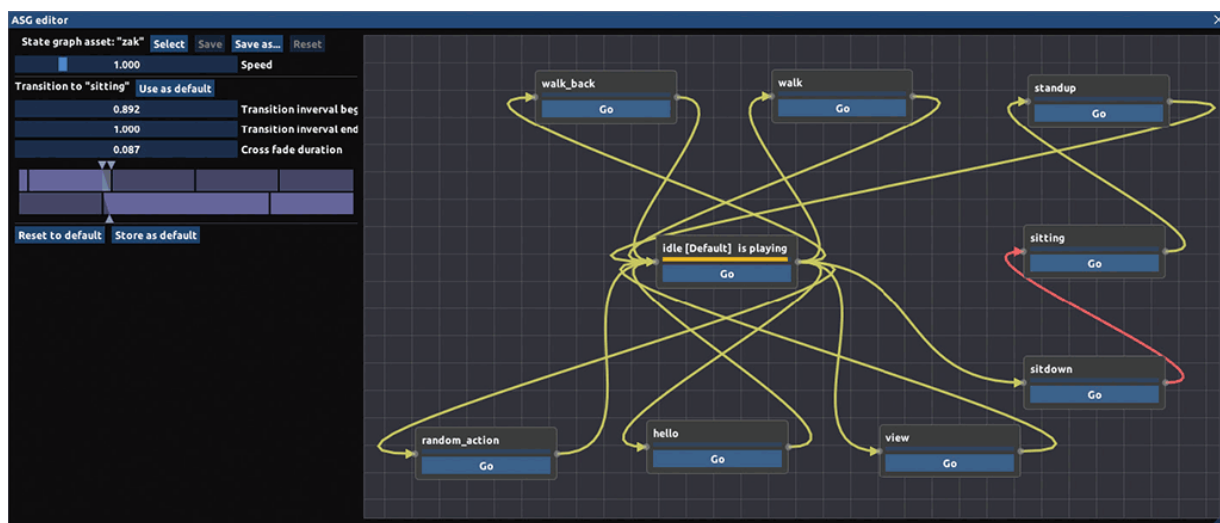


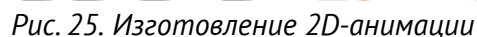
Рис. 24. Список анимаций в виде графических связей и состояний объекта из движка

В случае анимации персонажей есть понятие **«СКЕЛЕТНАЯ АНИМАЦИЯ»**. У персонажа есть наборы костей и суставов, которые определяют движения разных частей его тела. Суставы – это шарниры, позволяющие автоматически создавать реалистичные сложные движения. У сустава есть максимальная гибкость (определенный рабочий ход), и игровой движок определяет, когда эта гибкость заканчивается, после чего начинает работать другой сустав.

ЛОКАТОРЫ – это места, где мы можем прикрепить к модели что-то внешнее, например поместить ракетный отсек в наш вертолет или вложить пистолет в руку персонажа.

Если в игре есть **РАЗРУШАЕМОСТЬ ОБЪЕКТОВ**, это тоже необходимо задокументировать. Например, при попадании ракеты в наш вертолет в определенном месте должен идти дым или должны проявляться другие повреждения. Следует указать локаторы, то есть

Другой вариант – поменять текстуры. Например, при повреждении наш вертолет меняет базовую текстуру на черную и побитую, появляются царапины. Еще можно менять геометрию: если мы хотим, чтобы при критических повреждениях у вертолета отваливался хвост, в нужный момент можно заменить модель целого вертолета на модель поврежденного.



Логика создания 2D-анимаций схожа, но методы обычно связаны не с алгоритмами костей и суставов, а создаются, например, через ролики: по аналогии с классической мультипликацией, когда движения записываются покадрово либо через спрайты. Если мы делаем платформер в духе Mario, нужно будет определить, как должна выглядеть статичная модель, какие состояния у нее есть, и проработать анимации движения.

Хотя 2D-графика проще в освоении, одинаковый объем контента рисовать придется дольше, чем в 3D.

ГОТОВЫЕ РЕШЕНИЯ

Для несложных игр возможно самостоятельное создание анимаций с нуля. Но для многих вещей есть уже разработанные технологии, доступные в самом игровом движке или в сторках к нему. Готовые анимации можно найти и у сторонних людей или компаний. Можно воспользоваться сервисом Mixamo, там собрано огромное количество готовых анимаций. А вот создать студию для Motion Capture (технология, позволяющая оцифровать движения реальных актеров) могут немногие, поэтому такие заказы при необходимости обычно отдаются на аутсорс.

ШАБЛОНЫ хорошо работают для создания персонажей. Куда эффективнее сделать абстрактную модель скелета и натягивать на нее разные «шкурки» и текстуры, чем создавать каждую модель отдельно. Игрок будет видеть разных персонажей, но анимация движений – одна и та же, сделанная для одного скелета; он может быть даже готовым, взятым из движка. Сверху надевается модель с правильными параметрами костей, если мы хотим, чтобы персонажи отличались друг от друга длиной конечностей. Далее покрываем персонажа необходимыми материалами. Если в игре предусмотрена система одевания персонажей, значит, у модели должны быть локаторы для шлема, наручей и других элементов одежды.

Существует довольно много генераторов моделей, позволяющих не задавать вручную все параметры (от роста до длины носа). Чтобы добавить персонажу индивидуальности, художник может разнообразить текстуры шрамами, татуировками и другими отличительными чертами.

Предметы и окружение чаще всего уже есть в библиотеках, так что всегда можно ограничиться созданием только материалов и текстур.

Чтобы игры, собранные из готовых ассетов, не выглядели банальными, разработчики стараются во всем придерживаться выбранной стилизации. Это выглядит красиво, скрывает использование базовых моделей и выделяет игру среди конкурентов. Вспомните *Deus Ex: Human Revolution*: разработчики выбрали золотисто-черную гамму цветов как отсылку в эпохе Возрождения и старались придерживаться этих, ставших фирменными, цветов.

Для реалистичной (даже стилизованной) игры действительно существует множество ассетов. При знании технологий и их подключения разработчик может сильно облегчить себе жизнь. Но часто разные модели сделаны в разной стилистике или с разным уровнем качества, поэтому сложно создать потрясающую графику, используя только готовые ассеты. А вот для прототипов и других не финальных версий игры такие модели – отличный вариант.

Добавление моделей (готовых или своих) требует определенной квалификации, нужно уметь их экспортировать и анимировать, прикреплять локаторы, визуальные эффекты и т. д. Для мультяшных игр моделей значительно меньше, зато создавать их сильно проще и дешевле.

ВИЗУАЛЬНЫЕ ЭФФЕКТЫ

Как правило, одними моделями дело не ограничивается. Если наш многострадальный вертолет должен уметь красиво взрываться, скорее всего, технически это будет реализовано с помощью VFX (Visual Effects). Для вспышек, обломков, падающих вниз, вспыхивающих синих глаз зомби, дымного света – всего, что в игре двигается, но не анимируется, обычно используют готовые решения внутри движка. Для 2D-графики это также актуально: если что-то с чем-то взаимодействует, можно дорисовать эффект.

ПАРТИКЛЫ, или система частиц, нужны, чтобы передать нечто, состоящее из множества частиц. Брызги крови или искры – классические примеры такой системы. Рисовать каждую искорку отдельно было бы неэффективно. Поэтому, если мы хотим передать искры от удара мечом по камню, нам нужно задать одну элементарную частицу (искорку), точку, где камень сталкивается с оружием, область распространения и закон эволюции. После чего определяем ряд параметров, отвечающих за поведение искр, таких как, например, направление и зона, куда клоны этой искры должны разлетаться от удара. В движках обычно есть готовые партиклы, но при нестандартном запросе их приходится разрабатывать отдельно.

Важно помнить, что создаваемые эффекты предстоит совместить со звучанием, и это не самая тривиальная задача – хотя бы потому, что звук имеет источник, громкость и другие свойства. Чаще всего студии берут звуки из готовых библиотек, где можно найти что угодно: от предсмертных криков до взрывов. Если есть необходимость, можно делать самостоятельные записи.

ОСОБЕННОСТИ РАЗРАБОТКИ МОДЕЛЕЙ ДЛЯ ИГР

Итак, мы создали вертолет, анимировали его, добавили текстуры, обозначили локаторы, где что взаимодействует, «прикрутили» все эффекты и добавили озвучку. Но игровая модель динамична, она взаимодействует с окружением и другими моделями, поэтому так важно заранее продумать все эти взаимосвязи, чтобы грамотно поставить задачу.

Следующий вопрос – **ФИЗИКА ВЗАИМОДЕЙСТВИЯ** объектов игрового мира. Допустим, мы обозначили, что в наш вертолет нужно попасть ракетой, чтобы его сбить. Так как мы сделали качественный, фотореалистичный, многополигональный вертолет, расчет столкновения с ним ракеты будет занимать слишком много вычислительных мощностей.

Здесь нужно упомянуть о коллизии. Термин «коллизия» уже по своему названию дает общее представление о том, что это такое. Английское слово *collision* переводится как «столкновение» и в целом достаточно точно описывает суть дела.

Все мы знаем, что в физике столкновение – это некий процесс, при котором происходит взаимодействие частиц или объектов. И если в реальности оно может быть обусловлено, например, действием электромагнитных сил электронов, то в играх это, очевидно, симуляция реального процесса. Как и любая симуляция, она может иметь разную точность и степень приближения к оригиналу.

У большинства может возникнуть впечатление, что коллизии – нечто, присущее в первую очередь трехмерным играм. Но все то же самое актуально и в 2D. Давайте посмотрим на примерах.

Допустим, вы играете в свой любимый *Overwatch* или *PUBG*. В этом случае вы управляете поведением некоего объекта, который визуализируется трехмерной моделью. Если это человек, то расчет движения его конечностей с учетом всех суставов и мышц представляет собой непростую задачу, так же как и пересечение его частей с другими моделями на экране. Загружать процессор просчетом таких коллизий было бы нерационально. В этой ситуации модель персонажа в расчетах упрощают до некой зоны пространства – цилиндра, шара или эллипсоида. И во время игры считается лишь приближение друг к другу цилиндров с заданным радиусом и известными координатами оси симметрии, находящейся в центре цилиндра. Такая геометрия выглядит гораздо проще.

Когда же дело касается 2D, все в целом имеет схожую логику. Иной может быть лишь реализация. Сравнение зоны вокруг объектов в тех же платформах может сводиться к сравнению пересечения прямоугольника с окружающей локацией. Это как зона кликабельности вокруг кнопки: мы ее не видим, но она есть. Такую зону иногда называют коллайдером. С физическими ускорителями у них общее только одно: и там, и там происходят столкновения.

Другой пример – это коллизии элементарных частиц, математических точек. Такое можно представить себе, посмотрев на шарики, которые летят в популярных бабл-брейкерах, арканоиды или расчеты попадания патрона в противника.

Для регистрации таких попаданий в игровую сущность добавляют так называемые **ХИТБОКСЫ**. В большинстве игр это просто невидимая коробка, грубо покрывающая модель снаружи и регистрирующая столкновение с другими игровыми сущностями. Это важно, потому что, например, стрелять, учитывая реальную геометрию, очень затратно.

Классическое дизайнерское решение: есть мускулистый воин и миниатюрная эльфийка. Очевидно, что попасть выстрелом в более крупный объект легче. В старых играх можно найти примеры, когда из-за разницы в хитбоксах имело смысл всегда играть за самого маленького персонажа. Если персонажи больше ничем не отличаются, чтобы эльфийка не имела геймплейных преимуществ, часто им делают одинаковые хитбоксы. Например, в шутере *Arx Legends* персонажи имеют разный размер хитбокса, поэтому те, у кого он меньше, получают повышенный урон; так компенсируется сложность попадания. Для генерации хитбоксов обычно используют готовые решения в 3D-редакторах.

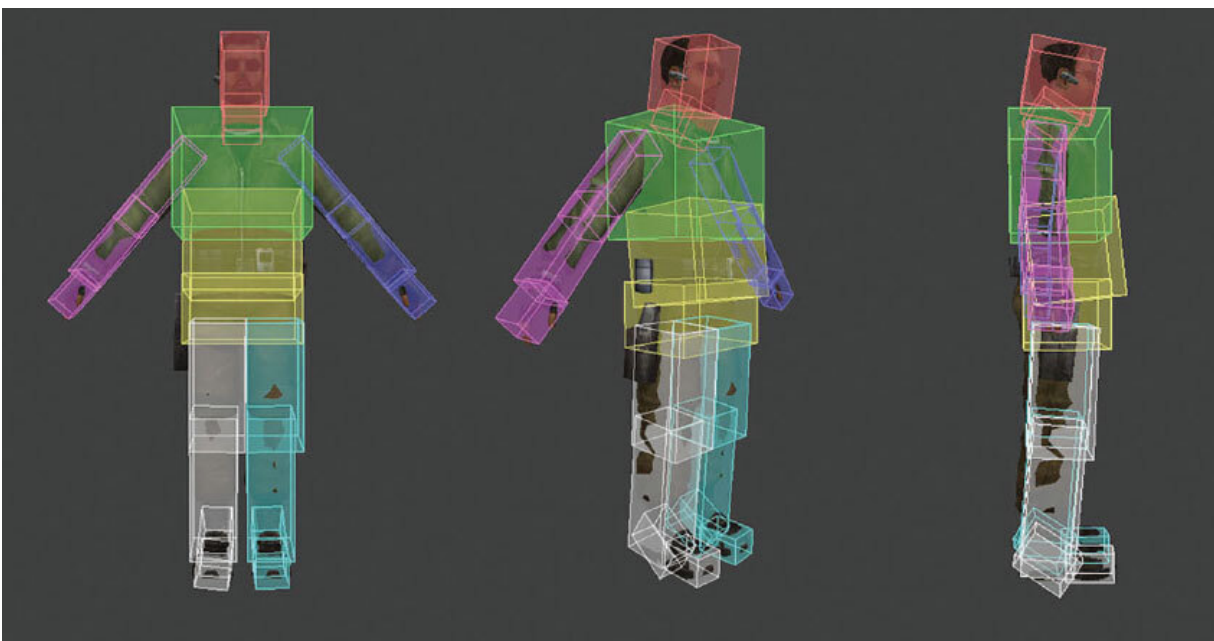


Рис. 26. Визуальный пример хитбоксов

Следующая непростая задача – реализация удара по объекту. Вы наверняка замечали, что во многих ММО, когда персонаж бьет мечом по своему сопернику, он даже не попадает по модели: просто машет оружием, а с противника списывается ХР. Дело в том, что адекватно показать удар холодного оружия по телу не просто. В жизни довольно странно представить себе ситуацию, что ты проткнул кого-то мечом, а он остался стоять, как стоял. Поэтому многие дизайнеры скрывают такое соприкосновение за столпом искр, чтобы не было заметно, что оружие не касается персонажа. Для огнестрельного оружия такой ширмой часто служит эффект разлетающихся брызг крови.

RAGDOLL – процедурный вид анимации в рамках физики, заданной в игровом движке, или же законов эволюции, которые написал программист, – при этом художник не задает параметров движения. Персонаж ведет себя как тряпичная кукла, которую уронили (отсюда и название). Теперь каждая кость и сустав отдельно подчиняются только физике. Представьте себе, что в персонажа выстрелили из оружия или он подорвался на mine. Пока у него есть

ХР и он жив, заданная анимация определяет, что будет происходить с его телом в этот момент. После смерти же он превращается просто в еще один физический объект, который может красиво отлететь в сторону или эпично врезаться в стену.



Рис. 27. Ragdoll

Есть программы для анимации одежды, волос, пролитой жидкости, ведь сделать ее для всех вариантов движения невозможно. Например, плащ не может просто висеть, его движение как минимум напрямую зависит от движений персонажа. Но даже такие готовые решения необходимо соединять со своими моделями, что уже требует определенной квалификации. Поэтому начинающие разработчики нередко предпочитают делать, например, пиксельную графику, чтобы не связываться со сложностями физики взаимодействия, рэгдоллами и прочими сложными сущностями. Тем не менее опытный художник в состоянии справиться со всеми этими задачами даже в одиночку.

ПОСТПРОЦЕСС, ИЛИ ПОСТОБРАБОТКА, – это метод обработки изображения, когда на уже готовую картинку с моделями накладывают определенные эффекты. Их очень много, и применяют

их для разных целей: например, мы хотим, чтобы готовые модели были стилизованы под рисунок от руки; если игрок едет на танке, можно создать ощущение, что он смотрит сквозь танковую оптику – с потертостями, определенным искажением картинки и так далее. Если игрока пугает монстр и экран в это время дрожит, это тоже постобработка, добавленная для атмосферы страха и напряжения. Или при тяжелом ранении экран игрока окрашивается красным, символизируя, что персонаж на грани смерти. Обычно наложение таких эффектов – задача технического художника, то есть специалиста, который является связующим звеном между 3D-художниками и программистами. Такой человек способен реализовать художественное видение в рамках технических возможностей движка.

ОСВЕЩЕНИЕ – одна из самых глобальных задач для настройки. Мы задаем источники света, а материалы моделей уже определяют, как падают тени, отражается свет и пр. Для большинства игр тени задаются процедурно, вручную никто не прорисовывает бесконечное количество вариантов. Освещение – очень ресурсозатратный процесс, особенно для маленьких объектов. Поэтому во многих играх тени можно отключать в настройках графики, чтобы иметь возможность играть на менее мощных компьютерах. В этом случае свет просто заменяется на проектор.

Иногда левел-дизайнеры точно знают, что некоторые предметы (деревья, памятники и т. д.) не будут двигаться. Тогда их обозначают как статические и просчитывают, как падает свет, только один раз.

Но свет может быть динамическим, а не статичным, и в этом случае нельзя просто зафиксировать тени за предметами. Если объект двигается или разрушается, значит, надо суметь просчитать эти изменения в динамике, и это одна из самых тяжелых для сцены задач. Добавить в сцену десять монстров может быть намного проще, чем одну дополнительную лампочку. Оценить стоимость объектов в кадре с точки зрения производительности вам могут помочь различные гайды на эту тему, а также профилирование:

в движке игры вы можете получить информацию, сколько в процентном соотношении стоит в данном кадре рендеринг моделей или освещения или расчет игровой физики.

Для мультяшной анимации не так критично, если тени падают как-то не так, а вот фотореалистичный объект с «кривыми» тенями сразу добавит вам комментариев про «игру из девяностых».

Нельзя не упомянуть об **ИГРОВЫХ ВИДЕОРОЛИКАХ**. В большинстве игр есть хотя бы начальная катсцена. Созданием таких роликов обычно занимаются отдельные специалисты.

Как правило, маленькие студии выбирают постановочные ролики, основанные на готовых решениях, предлагаемых игровым движком. Очень немногие студии делают качественные CGI-ролики^[74] самостоятельно, так как это очень трудоемкий и дорогой процесс. CGI-ролики генерируются не в реальном времени, поэтому можно выделить куда больше ресурсов на обработку каждого кадра, и картинка получается по качеству выше, чем при реалтайм-рендеринге в игровом движке. Для них требуются отдельные модели и ассеты, а стоят такие ролики сотни тысяч долларов.

Ролики, созданные с помощью движка, или CGI-ролики, у более богатых товарищей могут быть использованы для продвижения, например, в сторсах. Для клиентских игр стало стандартом делать красивые тизеры и трейлеры, так как людям скучно смотреть только скриншоты, видео вызывает куда больше эмоций и интереса. Для их создания покадрово составляется сценарий ролика. Важно при этом не забыть подогнать происходящее на экране под тайминг озвучки.

АРТ НА АУТСОРСЕ

Отдельная большая тема – это работа над созданием арта посредством **АУТСОРСА**. Внештатным сотрудникам труднее отследить изменения в проекте, поэтому важно быть особо внимательным при составлении ТЗ. Хорошим правилом все же считается хотя бы один

раз сделать арт своими силами и только потом отдавать на аутсорс, уже имея примеры, образцы и четкое понимание того, как создается качественная графика, сколько времени это занимает, и прочих нюансов.

В штате может вообще не быть художников, но вам все равно необходим опытный человек, знающий, что такое хороший арт-стиль, как построить процесс работы, и разбирающийся в технической стороне создания графики. В идеале он должен еще и уметь работать с гейм-дизайнерами: правильно понимать их идеи, при этом отстаивая при необходимости свое видение, что подойдет проекту, а что нет. В противном случае возникает распространенная ситуация, когда хороший художник просто не знает технической стороны вопроса; тогда предпочтительнее, чтобы человек, лучше всех разбирающийся в движке, взял на себя обязанность интегрировать получившийся арт в игру (в идеале это отдельный человек – технический художник).

Технические специалисты часто определяют успех ряда проектов в целом, потому что нужно, чтобы кто-то еще на этапе набросков игровых сцен мог дать оценку, с помощью каких технических средств можно достичь необходимого уровня качества картинки при минимальных ресурсах. Применять все средства оптимизации нет смысла, это никогда не окупится. Поэтому приоритеты должны быть расставлены в самом начале проекта.

Таким образом, успешная разработка графического контента определяется тремя важными аспектами.

Во-первых, это базовый выбор самого дизайна (как будет выглядеть наша игра), а также технические средства, с помощью которых мы будем достигать необходимого качества картинки при имеющихся у нас ресурсах. В случае правильного выбора создание нужных моделей не выйдет за рамки бюджета и сроков, а для достижения результата будут использованы именно те технологии и инструменты, которые подходят конкретной команде и конкретному проекту.

Второй важный момент – правильная постановка ТЗ. Грамотно составленное техническое задание обычно получается довольно громоздким, ведь исполнители должны иметь четкое представление о создаваемых объектах, их анимациях, взаимодействиях, локаторах, эффектах и пр., а также понимать конкретные сроки выполнения работ.

И третья необходимая вещь: какими бы квалифицированными ни были ваши специалисты, им все равно нужен менеджмент, чтобы нормально построить процесс работы и избежать ситуации, когда, например, вся команда ждет работы одного художника.

Три столпа успешной разработки графики: удачный базовый дизайн, конкретные задачи и качественное исполнение.

Сюжет и нарратив

В игры играют. Их не смотрят, как кино, не читают, как книгу. Текстовые игры больше не могут рассчитывать на признание широкой аудитории, поэтому дизайнерам необходимо задумываться о других методах вовлечения в игровой мир.

Стоит отметить, что иногда хорошая история сама по себе, особенно в одиночных приключенческих играх, может вовлекать не хуже, чем игровые механики, и некоторые игры создаются именно исходя из сюжета.

Но не для всех игровых жанров это актуально. Сеттинг игры, будь то технологичное будущее на другой планете или безбрежная пустыня – обязательная составляющая игрового мира. Сюжет же, история в выбранном сеттинге, может быть необязательным компонентом, но часто помогает добиться целей гейм-дизайнера.

Основная история может развиваться отдельно от геймплея, то есть игрок решает головоломки, побеждает в битвах и таким образом продвигается по сюжету, но при этом не участвует в событиях. Обычно такой геймплей характерен для простых мобильных игр. Ярким примером может служить баттлер *Mighty Party*, где есть игровое меню с отдельным разделом «Приключение». Это самое «приключение» и есть сюжетная линия для данной игры, своего рода «одиночная кампания».

Игровой сюжет – это события, связанные в единую последовательность. Он обычно сильно зависит именно от действий игрока. В MMORPG или AR-играх (играх с дополненной реальностью) типа *Pokemon Go* участники могут находиться на разных точках сюжета, но при этом взаимодействовать друг с другом на общем игровом поле. Сюжет – это не только непосредственно игровые события; он может определяться просто логикой повествования, например давать предпосылки к стартовой точке геймплея.

Сценарий же – это уже подробно прописанная последовательность действий, которую должен выполнить игрок, чтобы достичь цели игры. В RPG-играх, например, одну и ту же задачу можно решить разными способами (убить или подкупить стражу, спрятаться или атаковать), а для квестов и визуальных новелл такая последовательность обычно жестко predetermined.

Общий сценарий предполагает цепочки связанных событий, которые приведут игрока к разгадке основных вопросов сюжета. Но игрок не обязан двигаться строго по сценарию; значительная часть геймплея может лежать в стороне от основной линии (крафт, фарм, выживание).

ЗАЧЕМ ИГРАМ НУЖЕН СЮЖЕТ

Во-первых, он помогает предположить игровой путь, задает цели. Марио должен найти принцессу, нужно вернуть магический артефакт, чтобы восстановить баланс Добра и Зла, и т. д. Сложный мир *Skyrim* или *World of Warcraft* без историй превратился бы в хаос множества игровых сущностей, в которых невозможно разобраться. Сюжет дает мотивацию совершать те или иные действия: мы не просто убиваем очередного босса, а спасаем Хайрул от Ганона, не просто катаемся на катере, а пересекаем радиоактивные каналы, чтобы добраться до Илая и Джудит Моссман.

Во-вторых, сюжет часто работает на погружение в сеттинг, помогает раскрыть игровой мир, сталкивая героя с теми или иными персонажами и игровыми ситуациями. Даже в простых играх, где нужно собирать в ряд геометрические фигуры, приятнее ассоциировать себя с садоводом, собирающим урожай, или дворецким, обустривающим особняк.

Запоминающиеся герои *Angry Birds* – уже не просто элемент игровой механики, они повышают вовлечение в геймплей, делают его уникальным. Казалось бы, простая игра тайм-киллер (от англ.

time killer – «убийца времени»), но на самом деле персонажи серьезно проработаны. Птицы – характерный образ, вспомнить хотя бы, на гербах скольких стран изображены эти гордые пернатые (США, Россия, Польша, Германия и другие). Нам легко принимать их за «своих», за положительных героев. Свиньи – напротив, могут ассоциироваться с грязью и зловонными лужами, помоями. И эти существа похищают... яйца, то есть нерожденных детей. Так внешне незатейливая история о злых птичках приобретает глубокий подтекст. Это и есть нарратив – визуализация смыслов.

Плюс история облегчает усвоение правил игры. Капитан корабля собирает команду и отправляется с ней за сокровищами, поселенцы закладывают новые города, вампиры восполняют свою жизнь за счет чужой: действия юнитов логичны, их легко понять и запомнить. Летающие существа могут перемещаться дальше, тяжелое вооружение замедляет персонажей; вам не нужно заучивать наизусть, как в шахматах, ходы и способности фигур, окружение и история делают игровые механики живыми и понятными.

В большинстве случаев игровой процесс все же важнее истории. Но вспомните, сколько часов мы посвятили истреблению полчищ безликих крыс, сбору неведомых трав и грибов для алхимиков или возврату владельцам их фамильных драгоценностей. Эти примитивные квесты, стимулирующие однообразные игровые действия, сегодня уже не так привлекательны для игроков и вызывают только раздражение. Напротив, продуманный сюжет может вдохновлять на многократное прохождение одних и тех же уровней или карт.

Но сюжет – не самоцель. Прежде всего следует посмотреть на игры конкурентов, есть ли там сюжет? Решение о том, нужна ли проекту история и насколько проработанной она должна быть, зависит от его особенностей – прежде всего жанра и ожиданий аудитории. Если аудитория казуальная или длина игровой сессии – несколько минут, возможно, стоит ограничиться только проработанным сеттингом. Раннеры, простые платформеры, фермы

и другие подобные игры обычно не предполагают, что люди будут тратить силы и время на запутанные перипетии сюжета.

Для проектов, где геймплей интересен аудитории сам по себе, например головоломок или стратегий, бывает достаточно обозначить основные цели игрока и правила игрового мира, а сложные истории будут только отвлекать. Но если игровой процесс однообразен, сюжет может оказаться мотивацией продолжать такую игру, добавляя смысл происходящим игровым событиям. Конечно, это лишь примеры. Существует множество игр, где уживаются запоминающийся сюжет и интересный геймплей.

Сложная история с яркими персонажами может стать сопутствующей картинкой, оживляющей геймплей, особенно если вы придумали оригинальный сеттинг. Например, альтернативная история *Red Alert* до сих пор вдохновляет поклонников на создание фанфиков, а по вселенной *S.T.A.L.K.E.R.* написано около 90 новелл. Это говорит о том, что сюжет – это не только происходящее непосредственно в игре, но и события, которые привели к стартовой точке геймплея, и даже события, которые остались за кадром, но определяются логикой сюжета.

Нарратив также сглаживает игровые условности (сохранение, цифры-эффекты и пр.). Хорошим примером сохранений, подкрепленных нарративом, являются защитные знаки из *Call of Cthulhu: Dark Corners of the Earth*, которые отпугивают чудовищ и создают безопасные области, которые напрямую ассоциируются с точками сохранения.

Если раскрыть историю персонажей, они перестают быть болванчиками с характеристиками, а становятся полноценными героями, как, например, в *Mortal Kombat*, *Dota*, *Overwatch* или *Prime World*.

ЛИНЕЙНЫЕ И НЕЛИНЕЙНЫЕ ИСТОРИИ

Сюжет игры называют линейным, когда есть определенная последовательность событий и игровой процесс не влияет на историю. Обратная связь возможна: например, с очередным поворотом сюжета могут вводиться связанные по смыслу новые игровые механики.

Линейные сюжеты игр больше всего похожат на классические истории книг, фильмов, сериалов и т. д., поэтому к ним легче применять инструменты вовлечения из других видов искусства. Если вы способны заставить игрока с нетерпением ждать развития сюжета, он может и не заметить, что за него все решили гейм-дизайнеры и сценаристы. Это же и главный недостаток – такое решение ближе к интерактивному кино, а не к играм.

Нередко за визуальные новеллы берутся начинающие разработчики и инди-команды, не имеющие впечатляющего бюджета. Из-за недостатка ресурсов в таких проектах реализуют инструментарий видимости выбора: игроку «как бы» дают в руки управление сюжетом, но от его выбора на самом деле мало что зависит. Задача нарратива в этой ситуации – замаскировать эту ловушку.

Предложив игроку точки выбора, вы можете добавить интерактивности. Сюжет может предполагать разветвление, когда после очередного решения история разворачивается так, а не иначе. Таким образом, ваша игра получит возможность перепрохождения и сможет заставить игрока чувствовать ответственность за свои решения. Но продумать придется очень многое; каждый такой выбор предполагает написание отдельной истории и создание соответствующего количества контента, при этом совершенно не факт, что игрок будет изучать все возможные варианты развития событий. Если игра предполагает выход DLC, все эти варианты нужно будет учитывать, чтобы не ломать логику повествования.

Чтобы облегчить работу, гейм-дизайнеры могут вводить несколько параллельных историй, ведущих к одному исходу. Недостаток такого решения в том, что игрок теряет ощущение управления игровыми

событиями, ведь что бы он ни выбрал, история в итоге вновь становится линейной.

Игры с нелинейным повествованием, когда игрок сам определяет, в каком порядке он будет знакомиться с игровыми событиями – серьезный вызов. Как мы писали, именно выбор действий игрока имеет ценность. Так что, с одной стороны, это решение будет способствовать большему вовлечению. Но, с другой, такие интерактивные истории, когда повествование переплетается с геймплеем, требуют огромного количества работы. Контент должен быть доступен игроку, только когда это логически обусловлено предыдущими игровыми событиями, так что предусмотреть все варианты будет очень непросто. Когда вы рассказываете несколько параллельных историй, участником которых может быть игрок, есть риск, что он не заметит их связи. С этой проблемой столкнулись дизайнеры многих RPG. И если в классических играх типа *Baldur's Gate* игроку было позволено проходить множество квестовых линеек, раскрывающих сюжет персонажей группы или двигающих общий сюжет, то современные игры в основном ушли от этого. Так, в *WoW Legion* игроку необходимо выбрать локацию, сюжет которой он будет сейчас проходить. Пройдя одну локацию, он выбирает следующую. Все они заканчиваются роликами, связывающими сюжет каждой в единое целое – поиски столпов созидания для защиты Азерота от вторжения Легиона.

А есть ли история, например, в *FIFA*? Конечно, есть! Каждый матч складывается по-разному: победный гол на последней секунде, досадное поражение из-за неточности вратаря, – игрок преодолевает трудности, и каждая партия – это уникальный опыт и впечатления. Историю своими действиями создает сам игрок.

Есть целый класс игр, в которых нет предусмотренного разработчиками сюжета, но есть средства для его создания, в том числе с помощью действий игроков. Подобные игры за счет рандомной генерации уровней, набора игровых механик и других приемов оставляют развитие сюжета игроку, но требуют

внимательной работы гейм-дизайнера, чтобы игровые события не превратились в череду неинтересных и не связанных между собой действий игрока.

ПОГРУЖЕНИЕ

В целом гейм-дизайнеру будет очень полезно изучить классические труды о драматургии и сценаристике. Аристотель в трактате «Поэтика» давным-давно прекрасно объяснил, как важно понимать структуру своей истории, когда события определяют друг друга, а не происходят случайно.

Спад напряжения – большая проблема для видеоигр. Например, иногда финальная битва выглядит блекло из-за того, что к этому моменту игры персонаж уже достиг максимального уровня и снабжен всем необходимым обмундированием, и победа дается ему слишком легко.

Полезно будет изучить и мифологические сюжеты, которые легко воспринимает наше сознание. Все сюжетные схемы и архетипы выросли из мифологии, так что хороший сценарист просто обязан разбираться в материале. Путь героя в «волшебной сказке»^[75] и другие древние мотивы отражают представления общества об этике и морали, показывают, как, добиваясь своих целей, герой может меняться, становиться более великодушным или ожесточенным, побеждать или проигрывать. Эти знания помогут увидеть общую картину, как строятся такие истории.

Писатели знают, что отождествление себя с персонажем книги помогает погрузиться в предлагаемый мир. Это делает происходящее реальным и вызывает сильные эмоции. То же верно и для игр. У игр есть даже больше инструментов для этого, ведь решения «персонажа» – это решения игрока.

Игроки часто обращают внимание на «проработанность мира». Например, истории жизни второстепенных персонажей, их реакция

на принятые игроком решения (в репликах, обстановке домов и пр.), отсылки к другим произведениям – все это делает игровой мир живым и наполненным. Потерять 20 % лучников в стратегии – неприятно и влечет за собой последствия. Но потерять, по сюжету игры, вашего пожилого наставника, который с первой минуты приключения был с вами, учил сражаться, рассказывал о мире (незаметно объясняя правила), – настоящая трагедия. Гейм-дизайнеры, умеющие пользоваться такими приемами, могут рассчитывать на совершенно другой уровень эмоционального вовлечения.

Герои должны влиять на игровую реальность. Эффект «бога из машины», когда происходящее объясняется не мотивацией, а чем-то сверхъестественным или просто нелогичным, разрушает связь пользователя и игрового мира. Игрок должен чувствовать, что он (то есть действия его героя/героев) определяет сюжет. Поэтому, зная особенности жанра, сеттинга и ожиданий аудитории, необходимо продумать, какие цели стоят перед игроком (героями), как он будет пытаться их достичь, какие препятствия встретятся на его пути, будут ли они менять мотивацию героя и сможет ли герой в итоге добиться того, к чему шел.

Чтобы игроку было от чего оттолкнуться, при знакомстве со своим героем на помощь часто приходят начальные катсцены. Там нет геймплея, и авторы могут спокойно ввести игрока в курс дела: что это за мир, кто его герой, какой у него характер, – чтобы в будущем предопределить решения играющего. Многие игроки считают появление красивой катсцены наградой за пройденные испытания. Но следует помнить, что в эти моменты игра перестает быть игрой, пользователь ни на что не может повлиять, так что к кинематографичным вставкам следует подходить обдуманно. Чтобы дать игроку первичную информацию, также используют общение с другими персонажами на небоевых локациях, туториал, текстовые вставки.

Гейм-дизайнеры могут добиться сопереживания герою (неважно, положительному или отрицательному), одарив его харизмой и по-настоящему сильным, самобытным и проработанным характером. Но есть игры, где герои, с которыми может ассоциировать себя игрок, намеренно не наделены ярко выраженными чертами: их редко показывают на экране, говорят они мало, намеками, давая игроку возможность «додумать» характер самостоятельно, чтобы было легче ассоциировать героя с собой, понимать его мотивацию и отыгрывать выбранную роль. Зачастую воображение игрока может заменять картинку. Такой прием был использован и в *Half Life*, и в *Dead Space*, и во многих других играх. Во многом они противоположны тому же «Ведьмаку», где предполагается отыгрывание вполне конкретного персонажа, историю которого пишет не игрок. И характер героя тоже определяет не он.

Стоит отметить, что нарратив (повествование) и сценаристика – это две разные вещи. Написать сюжет для игры – это одно, и совсем другое – определить, какими методами дизайнер доносит до игрока всю полноту игрового опыта. Нарратив – не история, а совокупность всех инструментов (от арта до баланса), передающих этот опыт, причем желательно индивидуальный для каждого игрока.

ИНСТРУМЕНТЫ НАРРАТИВА

Ваша цель – не допустить нелогичных связей игровых событий и сущностей, но показывать их все вовсе не обязательно. Это сложная задача – определить, сколько информации вы можете дать игроку, чтобы он все понял и не запутался, но при этом мог бы и сам поучаствовать в создании собственной истории.

Классический пример – как показать, что персонаж умер? Можно передать сообщение через речь героев («Вчера были на похоронах Джона...»), можно вставить сценку (грустные люди мокнут под дождем, стоя над разрытой могилой под скорбный голос

священника, читающего молитву), а можно просто показать комод с несколькими фотографиями (герой идет в первый класс, оканчивает университет, женится на красивой девушке, вот его семейное фото с ребенком и последняя фотокарточка, самая большая, где он в военной форме, а само фото обрамлено траурной ленточкой).

Нередко нарратив подается через диалоговые фразы, дневники персонажей, записки. Но подготовка текстов – только одна из задач нарративного дизайнера. Не только они раскрывают мир игры. Нарратив *World of Warcraft* – не только диалоги и квесты; есть внутренние инструменты (дизайн окружения, абилки разных рас и классов персонажей, игровой интерфейс, пасхалки и пр.) и внешние (красивые дорогие видеоролики, рекламные арты и баннеры, книги о создаваемой вселенной и пр.).

Нарративный дизайнер – все еще гейм-дизайнер, он должен не только отстаивать интересы истории, но и разбираться в игровых механиках проекта. Игровой нарратив (в отличие от сценария) не может существовать в отрыве от геймплея.

Например, начало игр про выживание почти всегда вызов для игрока. Нужно собирать ресурсы, крафтить оружие и одежду, бороться с агрессивными животными, строить убежище и т. д. Логично, что персонаж оказывается в незнакомом и таинственном месте (в лесу, в пустыне); у него нет навыков, поэтому он может пользоваться только примитивными орудиями; окружение вызывает чувство опасности и тревоги. Сам жанр определяет, какие игровые и нарративные механики должны быть использованы, чтобы передать конкретные ощущения от игры. Это может быть и чисто физическое действие: например, чтобы разрубить арбуз в мобильной игре, можно предложить игроку не нажимать на оружие, а пальцем провести по экрану, дав почувствовать, что удар был совершен собственной рукой. Если дизайнер умеет наделять игровые механики новым смыслом, можно сказать, что он – хороший нарративщик.

Саунд-дизайн часто используется как часть нарратива. Звуки природы или города, наслаивающиеся друг на друга, создают ощущение реального, живого мира. Озвучка помогает углубить погружение, создает необходимое настроение и атмосферу, а также подчеркивает геймплей. Например, в *Last of Us* звуки, издаваемые зараженными, зависят от степени их заражения. Плотный звук *God of War* (2018) помогает почувствовать силу и мощь персонажа и окружения. Особенно хочется отметить *Hellblade*, создатели которой в начале игры не зря рекомендуют подключить наушники. Разработчики подошли к саунд-дизайну с особенным вниманием: используя технологии в том числе бинаурального звука, они добились удивительного эффекта присутствия. В голове главной героини существуют несколько голосов, общающихся друг с другом, и то, откуда идет звук, помогает определить, например, с какой стороны ждать нападения. Звуки, которые слышит героиня, слышит и игрок, тоже как будто внутри своей головы. Тем самым подчеркивается безумие Сенуа, создается максимальное погружение и чувство тревоги.

Использовать инструменты других видов искусства, чтобы вовлечь или вызвать эмоции, можно и нужно. Это могут быть угол камеры, музыка, сюжетные повороты и так далее. Но больше всего игроки ценят свой собственный и неповторимый опыт от игры, поэтому нельзя лишать их возможности самостоятельно принимать решения. Каким бы гениальным и разветвленным ни был ваш сюжет, именно интерактивность, отзывчивость мира на наши действия и решения, делает игры столь привлекательными. Так что основная цель дизайнера – раскрыть мир и сюжет с помощью игровых механик и окружения. В отличие от катсцен или текста, их нельзя пропустить или отмахнуться, они то, зачем игрок пришел в игру. Скорее всего, в будущем цениться станут именно игровые истории, переданные через геймплей.

Нам нужно, чтобы игрок взаимодействовал с миром так, а не иначе, но при этом не терял ощущения контроля над ситуацией, о

котором мы говорили выше. Здесь нам очень помогает левел-дизайн: например, указывающие на игровые события надписи на заборах, нагромождения личных вещей в заброшенном доме, единственный освещенный участок улицы, кажущийся безопасным, разрушенные здания, намекающие игроку, что город переживает не лучшие времена. Левел-дизайнеры называют это «историей через окружение». Здесь нет влияния на геймплей; игрок может не обращать внимания на детали, а просто бежать дальше; но дизайном заложена возможность познакомиться с событиями игрового мира, в том числе через окружение.

Так называемые пасхалки как часть истории – это также способ общения разработчиков со своими игроками. Предметы в *World of Warcraft* могут намекать на развитие игры в будущих обновлениях, а название книги в «Ведьмаке» может объяснить политику студии относительно DRM^[76]. Нарратив во многом схож с UX, он также незаметно вплетается в игру, подчеркивая игровые механики, соединяя разные части игры.

Итак, знания о том, как пишутся сценарии для книг и фильмов, могут быть полезными, однако слепо копировать их не получится – все же игры имеют свои законы. Игровой нарратив – это инструмент для вовлечения, создания условий сотрудничества между игроком и игровой реальностью.

Как и во всех остальных элементах игры, очень важно понимать, зачем вводится та или иная нарративная составляющая, какие новые ощущения она подарит игроку и, главное, как она соотносится с уже принятыми игровыми механиками и другими гейм-дизайнерскими решениями. Если у вас в голове есть гениальный сюжет для игры, это замечательно, но недостаточно. Плохо сбалансированную и скучную стратегию не спасет никакой сюжетный поворот. Верно и обратное: сырая, нелогичная история может испортить впечатления от самого проработанного геймплея. История – лишь один фрагмент пазла, который вам необходимо собрать, чтобы получить хорошую игру.

Подготовка к релизу

Когда предыдущие этапы пройдены, процессы по производству контента налажены, можно переходить к следующему этапу – тестированию и подготовке к релизу. На этой стадии проекта мы будем:

- шлифовать игру до финальной версии;
- оптимизировать ее, если необходимо, под разные платформы;
- проводить все маркетинговые активности и размещать игру в сторах.

Этот этап может занимать разное количество времени в зависимости от особенностей проекта: например, оптимизация мобильных игр под различные устройства Android вполне может растянуться на несколько месяцев. Все материалы для маркетинга следовало подготовить заранее, за несколько месяцев до релиза, чтобы спокойно следовать составленному маркетинговому плану.

Для некоторых проектов маркетинг занимает даже больше времени, чем непосредственно разработка продукта. В этой книге мы не говорим подробно об игровом маркетинге. Это широкая тема, которую мы детально раскроем в отдельной книге.

Чем ближе к релизу, тем больше времени разработчики уделяют тестированию. Конечно, на всех этапах производства игры плейтесты используют как один из главных инструментов для подтверждения гипотез. Теперь мы переходим к другим типам тестирования для выявления недоработок/багов.

Особенности проекта могут диктовать разные подходы к тестированию. Если вы работаете над игрой самостоятельно, то, естественно, главным тестировщиком проекта будете вы сами. Никто лучше вас не знает проект, но процесс этот довольно трудоемкий – стоит отвести на него побольше времени.

Также настоятельно рекомендуется прибегать к помощи других специалистов, искореняя субъективность видения собственного

проекта и «идеальности кода», когда «тут все нормально, ничего тестировать не надо». Со временем взгляд на свой продукт замыливается, поэтому имеет смысл еще на ранних этапах производства показывать игру другим гейм-дизайнерам (на конференциях, конкурсах и других площадках), чтобы получить отзыв от профессионала, способного предложить лучшие решения.

Обычная практика – обратиться к друзьям и знакомым, готовым уделить время вашему проекту. Если это опытные геймеры со стажем, тем лучше для вашей игры. Но не забывайте, что они – необязательно ваша целевая аудитория. А чтобы проводить объективное тестирование, особенно с целью проверки, насколько интересен предлагаемый геймплей, очень важно подобрать правильных людей.

Чем занимается тестировщик и виды тестирования

Первая цель хорошего тестировщика – не дать выпустить проект в недостаточном качестве. В большинстве западных студий, если есть зафиксированное обоснование, QA-специалист имеет право запретить запуск конкретного билда или проекта, и только высшее руководство студии может отменить это решение. Понятие QA включает в себя тестирование, но не ограничивается только им. Под тестированием традиционно понимается поиск дефектов (багов), QA же занимается и оценкой рисков, и анализом требований, и обеспечением всего процесса тестирования.

Когда тестировщик мало пересекается с гейм-дизайнерскими идеями, основная задача – проверить, выполнены ли все работы в соответствии с требованиями визионеров и создателей, подтвердить теории и опровергнуть опасения заказчика. Обычно тестировщик работает с некой сущностью: он должен убедиться, действительно ли она реализована исполнителями качественно, соответствует ли ожиданиям заказчика и здравому смыслу в целом.

Хорошая практика, когда любой исполнитель – первый тестировщик своей работы. Каждый должен протестировать ее результаты сам, прежде чем показывать другим сотрудникам.

Цели тестирования могут отличаться на разных этапах жизни проекта. Во время активной разработки важно вызвать как можно больше «отказов», чтобы выявить скрытые в программе дефекты. Поэтому по большей части будет проводиться «негативное тестирование», цель которого – пытаться всеми способами «сломать» игру, чтобы впоследствии устранить найденные дефекты. Ближе к релизу (не только всего проекта, это может быть и релиз части функционала) уже проводятся приемочные тесты, цель которых – доказать, что все работает как задумано.

Первое, над чем работают QA, – **SMOKE-ТЕСТЫ**. Это быстрая проверка работоспособности текущего билда: что он вообще запускается, что можно зайти в игру, создать аккаунт и пр. Задача smoke-тестирования – быстро удостовериться, что продукт функционирует, базовые функции работают и последние изменения не конфликтуют с текущей версией игры. Если проблемы начинаются уже на этом этапе, проверять что-либо дальше не имеет смысла, можно прекращать тестирование и писать разработчику, что требуется новый билд.

Плохая практика, если постановщик задачи не хочет даже открыть то, что он предоставляет для тестирования. Запустить билд, открыть страницу сайта, скачать мобильное приложение может любой, даже не специализирующийся на тестировании сотрудник. Тестировщику лучше не тратить на это время, а сосредоточиться на проверке, действительно ли продукт работает так, как задумано.

Убедившись, что ваш билд функционирует, можно переходить к **ТЕСТИРОВАНИЮ ЗАДАЧ**. Базовая логика тестирования – сделать так, чтобы игра работала по заявленным требованиям. Здесь все понятно: если гейм-дизайнер обозначил условия для игрового события, QA должен проанализировать и сверить полученную информацию с той, что была заявлена, и проверить, не нарушает ли она общих стандартов функционирования продукта. Такая работа занимает 70 % работы QA-специалиста: он проверяет, что конкретная задача действительно выполнена и в полном объеме соответствует описанию. Поэтому чем больше информации тестировщик получит от заказчика, тем лучше будет результат.

Не стоит чего-то утаивать: напротив, нужно дать максимум вводных данных, чтобы специалист мог верно оценить объем тестирования, а не тратить время на «найди то, незнамо что». Чтобы это работало эффективно, разработка должна вестись в рамках конкретных задач. Например, если вы используете Trello, можно перенести задачу в колонку Need test, откуда QA-специалист, в свою

очередь, либо отправит ее в Done, либо сообщит об обнаруженных дефектах через баг-репорт и отправит карточку на доработку.

В классической разработке программного обеспечения тестировщик не проверяет что-то вне технического задания. Но игровое тестирование предполагает еще и проверку здравого смысла тех или иных решений, особенно там, где спецификации нет. Простой пример: если специалист по тестированию замечает, что в нашей условной игре у игрока есть возможность спрятаться за камень или другой объект, чтобы безопасно уничтожать монстров или живых противников, которые не могут зайти в эту зону, он должен обратить на это внимание других членов команды.

Есть еще **РЕГРЕССИОННОЕ** тестирование, направленное на поиск ошибок в уже проверенных участках кода. В любом проекте есть какое-то слабое, проблемное место. Как известно, баги – сущность живучая, поэтому нужно убедиться, что не вернулись ошибки, которые были исправлены ранее. Предположим, нам стало известно о некой архитектурной проблеме, и в нашем шутере игроки постоянно проваливаются под землю. Даже если программисты на определенном этапе смогли исправить этот дефект, важно зафиксировать и определить его ключевые детали и отправить на регрессионное тестирование, то есть добавить в базовую проверку последующих версий игры, ведь такая проблема серьезно влияет на геймплей. Это позволяет заранее предупредить команду о возвращении бага, а не ждать негативных отзывов от игроков.

Это могут быть и абсолютно новые дефекты – ключевое значение имеет то, что они появились в ранее проверенной и исправно работающей части программы; такие баги могут возникнуть при любом изменении кода. Название «регрессионное» такое тестирование получило, так как в ходе него проверяют, не стала ли система хуже, не «регрессировала» ли она.

Чек-листы и тест-кейсы можно составлять и до появления полной версии игры. Для составления некоторых из них достаточно только утвержденных спецификаций. Например, если у нас уже есть

готовый макет главного экрана или точный перечень способов оплаты игровых товаров, вполне можно написать кейсы на проверку всех кнопок экрана или платежных систем. Однако, чтобы их использовать, придется подождать рабочий билд.

ГЕЙМПЛЕЙНЫЕ ТЕСТЫ организуют для самих разработчиков, директоров, журналистов и т. д., чтобы те оценили игровые механики. На любом этапе разработки важную роль играют регулярные внутренние совместные плейтесты. Когда вся команда собирается вместе, чтобы поиграть в свою игру, шанс того, что кто-то перестанет понимать, что она собой представляет, сильно уменьшается. Такие встречи мотивируют дальше работать над проектом, ведь все видят результаты собственного труда, могут сразу заметить ошибки, обмениваться идеями и мнениями.

ФОКУС-ГРУППЫ – это специально подобранная аудитория, которой мы демонстрируем игру для проверки, насколько выбранным людям понравится то, что мы предлагаем. Скажем, мы хотим показать механику сражений на мечах историческим реконструкторам или понять, насколько интересен наш симулятор свиданий молодым девушкам из Санкт-Петербурга. В полученных отзывах о проекте кроется и обратная сторона: если результаты неутешительные, есть соблазн объяснить неудачу неправильным подбором фокус-группы. Это опасный подход – нужно быть абсолютно уверенным, что проблемы не в самой игре. Такой вид тестирования проводят на ранних этапах производства.

Полноценные **ИГРОВЫЕ ТЕСТИРОВАНИЯ НА ЖИВЫХ ИГРОКАХ** (альфа, бета, soft launch) покажут, насколько гейм-дизайнерские решения приятны для более широкой аудитории. Здесь важно именно собирать отзывы. Поиск багов лучше проводить раньше и предоставить его профессионалам; они сделают это быстрее и качественнее, чем игроки.

А/Б-ТЕСТИРОВАНИЕ предполагает, что мы делим игроков минимум на две группы и каждой демонстрируем отличную от другой группы некую новую игровую сущность. Суть такого

тестирования в том, что контрольная группа сравнивается с набором тестовых групп, где один или несколько показателей изменились. Так можно проследить, какие из изменений улучшают целевой показатель. Проверять геймплей таким образом очень сложно: если одной группе дать шанс урона X2, а другой нет, ощущения от игрового процесса будут слишком разными, чтобы сделать корректные выводы. Проверять, как работают разные монетизационные предложения (скидки, акции и пр.), напротив, очень эффективно.

Другой вариант: мы преподносим новый функционал постепенно: сначала, допустим, для 5 % игроков, потом для 20 % и т. д. При таком подходе, если на первых этапах мы выявляем какие-то проблемы, у нас есть возможность исправить их до того, как эти изменения затронут широкую массу игроков. Особенно это актуально для мобильных игр, здесь важно постоянно снимать метрики и проверять монетизационные схемы. Технически такой процесс не самый простой, так что в основном этим занимаются крупные игровые студии, для которых цена ошибки слишком высока.

Работая над некоторыми видами проектов, нужно быть морально готовым к тому, что тестирование выявит критические проблемы, способные отбросить проект назад на этап производства.

КОГДА ВСЕ ФИЧИ РЕАЛИЗОВАНЫ

Итак, мы находимся на этапе, когда все фичи в игре реализованы. Может быть, пока еще остаются проблемы и баги, но весь функционал работает, так что можно приступать к тестированию. Создается отдельная ветка, где начинается работа по устранению недочетов. Когда устранены самые критические дефекты и состояние билда доведено до допустимого уровня качества, эта версия может считаться стабильной и «замораживается». Теперь без

веских причин мы не добавляем в игру ничего, что может повлиять на качество, зафиксированное ранее.

Теперь важно провести **КОМПЛЕКСНОЕ ТЕСТИРОВАНИЕ СОВМЕСТИМОСТИ**, которое включает в себя тестовое покрытие^[77] на целевом железе. Обычно игровые студии обращаются к специализированным компаниям, особенно для игр под Android: целевых устройств очень много. ПК-игры тоже необходимо протестировать на машинах с разной производительностью, линейками видеокарт, оперативной памятью, разрядностью операционной системы и прочим. Проводить такие плейтесты самостоятельно может быть очень дорого; необходимо обратиться хотя бы к друзьям и знакомым, чтобы убедиться, что на разных устройствах все работает как задумано.

Даже простейшая минималистичная инди-игра может быть ненадлежащего качества и нуждаться в оптимизации.

Если в игре есть онлайн-составляющая, производится тестирование серверной части: замеряется нагрузка, проверяются возможные проблемы с соединением, безопасностью и защитой.

Следующий процесс – формальные приготовления для размещения проекта на игровых платформах. Это относительно просто для мобильных сторов (Google Play и App Store) и Steam, сложнее для Epic Games и очень сложно для консолей, у которых самый серьезный контроль качества перед размещением на их площадке.

Для предварительного тестирования необходимо хотя бы 10 человек, для ЗБТ^[78] или soft launch – от тысячи. Желательно, конечно, чтобы это были люди, входящие в состав целевой аудитории, подобранные с помощью таргетинга^[79]. Если в игру, направленную на женскую аудиторию, с помощью целевого трафика «загнать» 90 % мужчин, результаты будут соответствующими. Трафик покупают либо на целевую аудиторию, либо на большую массу пользователей, но раздробленную по каким-то признакам, – чтобы снять метрики со всех категорий игроков. Консольные игры больше

продвигают за счет бренд-маркетинга и PR, чем за счет трафика. ПК-игры где-то посередине: можно покупать трафик, но все равно важен органический трафик – то есть пользователи, пришедшие из поисковых систем.

Много внимания уделяется сбору метрик и отзывов. Гейм-дизайнер и продюсер должны заранее продумать, какие именно данные необходимы, чтобы сделать выводы и договориться с программистами о технической реализации сбора статистики. Современные технологии, например облачные, позволяют собирать и хранить большой объем данных, но это не всегда дешево. Для небольших проектов фиксация всех состояний и действий игрового монстра против игроков может быть неоправданным решением, хотя очевидно, что анализировать такое взаимодействие полезно. Можно фиксировать, например, не каждый факт смерти игрока от моба, а каждый десятый.

Еще на этапе составления вижн-документа за счет SWOT-анализа или других инструментов определяются основные риски проекта, в том числе спорные гейм-дизайнерские решения, которые могут отпугнуть игроков. Поэтому прежде всего нужно предусмотреть механизмы, способные на этапе тестирования отследить реакцию пользователей на то или иное экспериментальное решение.

Так же стоит поступать и для проверки выбранных монетизационных моделей: без статистики и метрик невозможно будет сделать выводы о том, что работает по гейм-дизайну, а что не работает вовсе.

Обычно после сбора информации аналитик формирует отчет, на основании которого продюсер или гейм-дизайнер может сделать выводы и внести необходимые изменения. Допустим, геймдизайнер предположил, что встреча с монстром, убивающим игрока четыре раза из пяти, должна стимулировать покупку специального меча. Цифры говорят о том, что это предположение оказалось неверным, никто не покупает это оружие, и задача гейм-дизайнера – понять, что пошло не так. Можно предположить, что игрокам 500 рублей

кажутся неоправданной ценой и они предпочитают на эти деньги купить себе кружечку хорошего пива. Другое предположение: пользователи воспринимают эту ситуацию как вызов, получают удовольствие от встречи с действительно сильным противником и не хотят упрощать себе задачу до нажатия одной кнопки. Гейм-дизайнеру предстоит обдумать все возможные варианты, предложить решения и после плейтестов снова проверить статистику.

Бывают и обратные ситуации, когда все работает не так, как запланировано, но получается все равно хорошо. Допустим, по каким-то причинам наш монстр убивает игроков не четыре раза из пяти, как было задумано, а девятнадцать из двадцати. Но пользователи не уходят: большинство действительно покупает предложенный меч, так что игра хорошо зарабатывает, а те немногие, кому удалось без доната справиться с мобом, чувствуют гордость и рассказывают своим друзьям о проекте, подарившем столько ярких эмоций. В этом случае баг превращается в фичу.

Отзывы игроков

Сбор обратной связи, безусловно, очень важная часть работы над игрой. Но нужно помнить, что далеко не все отзывы могут быть полезны. Например, на форуме, посвященном вашей игре, какой-то товарищ пишет, как ужасна ваша игра, что он не играет в нее уже год и не будет играть, пока «тупые разрабы» не исправят ту или иную ситуацию. Здесь есть два варианта: либо он все-таки проявляет активность в игре, ведь он пришел на этот форум, либо не играет и вряд ли будет. В любом случае его отзыв не поможет сделать игру лучше. Слушать ваших игроков, изучать их отзывы, общаться с ними во всех возможных каналах связи, чувствовать душу вашей игры и знать, что хотят игроки, это очень важно для того, чтобы привести игру к успеху. Но делать это необходимо осмысленно, отталкиваясь в первую очередь от аналитики, методологии разработки игр и здравого смысла.

Люди часто пишут негативные отзывы, не только сталкиваясь с проблемами и ошибками, но могут быть и просто «троллями», или не любить конкретный сеттинг, или иметь личный негативный опыт, связанный с вашей игрой, – вариантов довольно много. Такие люди зачастую самые громкие, поэтому просто по количеству постов или экспрессивности выражений делать выводы не стоит. Негативные отзывы вообще оставляют куда охотнее; когда людям комфортно, они редко вспоминают о возможности оставить свое мнение.

Есть и обратная проблема: людям действительно плохо, но они об этом нигде не пишут. Может быть, им неприятно признавать свои игровые поражения, или им просто лень писать отзывы, или, что еще хуже, они чувствуют, что есть проблема и им что-то не нравится, но не могут сформулировать, что же с нашей игрой не так.

Статистика и метрики реальны, и, если данные собираются корректно, можно увидеть настоящую картину происходящего на инфографике и сделать соответствующие выводы. Тем не менее

игры – это развлечение, и даже при великолепных цифрах игроки могут толпами покидать игру, потому что им неинтересно. Поэтому отзывы в сторах и на форумах могут не совпадать с данными статистики.

Игроки не обладают полной картиной того, что происходит в игре; более того, эта картина может быть просто ложной. Они могут считать, что какой-то класс сильнее всех, просто исходя из своего игрового опыта или потому что один из блогеров заявил об этом на стриме. На самом деле этот класс по характеристикам может быть слабее остальных, и если вы будете его улучшать, то с точки зрения игроков получится, что «тупые разрабы бафают имбу».

Другое дело, если некоторые непопулярные решения принимаются вынужденно, просто чтобы не сломать внутриигровую систему, например игровой баланс. В некоторых МОБА-играх разработчики ослабляют самых сильных и популярных героев, чтобы уравнивать всем шансы на победу. Для киберспортивных продуктов забота о балансе особенно важна. Но исчезновение отработанной тактики в *Heroes of the Storm* или ослабление любимого героя *League of Legends* вызовут серьезное разочарование у части игрового сообщества.

Однопользовательские игры могут быть в чем-то даже сложнее для восприятия сообществом. Первое, что вызывает ненависть, – это продажа продукта, не соответствующего ожиданиям игроков. Если вы обещали уникальный саундтрек, ваши трейлеры демонстрировали разнообразные механики и геймплей, а в итоге игроки не видят ни того, ни другого, отрицательных отзывов не избежать.

Если люди привыкли к чему-то, а получают другое, это тоже обманутые ожидания. Это большая проблема для игровых серий и франшиз: разработчики стараются внести что-то новое, улучшить свою игру, а игроки просто не готовы к этим изменениям. Часто это связано с тем, что неверно была определена целевая аудитория. Такие несоответствия ожидания и реальности могут стать пятном на

репутации разработчиков, но проект при этом может быть вполне успешным с точки зрения бизнеса.

Второй любимый негативный комментарий – «забагованная» или «сырая» игра. Нишевая аудитория в этом плане куда более лояльна: игроки готовы искать решения, ставить дополнительный софт и прочее, чтобы игра хотя бы запустилась. Аудитория массмаркета, скорее всего, не будет даже пытаться.

Третья проблема не такая глобальная, но тоже встречается. Это ненависть сообществ. Отсутствие чернокожих рыцарей в *Kingdom Come: Deliverance* было достаточным основанием к призыву бойкотировать эту игру. Другое дело, что аудитория игры была все-таки более нишевой и для этих людей реалистичный сеттинг и историческая достоверность происходящего были одними из главных преимуществ, так что никто не пострадал.

А вот если обидеть свою целевую аудиторию, это может стать серьезной проблемой. Если человек, представляющий компанию или продукт на рынке, позволил себе высказывание, оскорбившее или задевшее определенную категорию игроков, люди могут начать массово покидать игру. Особенно это частое явление для западной аудитории, в большинстве своем считающей, что если во главе компании стоит человек с определенными взглядами, которые не совпадают со взглядами современного общества, то ничего достойного эта студия все равно не сделает.

Заключение

После запуска

Для buy-to-play игр менять гейм-дизайнерские решения после запуска бывает уже поздно – остается только работать над патчами, пытаться исправить отдельные проблемы.

Финансовый провал сильно отличается у проектов на разных платформах. Например, для мобильных игр это ситуация, когда сумма, потраченная на привлечение пользователей, оказалась выше суммы заработка; или никто не покупает нашу клиентскую игру. Выбраться из такой ситуации довольно сложно, хотя некоторым играм это удавалось. Дешевого решения не существует, и выбор невелик: либо принудительно сворачивать работу над игрой, признав ее неудачной и стараясь отбить хотя бы что-нибудь, либо срочно вложить дополнительные ресурсы в надежде исправить положение.

К сожалению, в большинстве случаев, если проект стартовал плохо, никакие усилия его не вытянут. Даже если это инди-проект, где никто никому ничего не платит, такая ситуация сильно бьет по морали, и удержать людей становится крайне сложно. Если гейм-дизайнер, любящий свой проект, берет на себя и функции продюсера, решение отказаться от игры как от провальной оказывается очень болезненным.

Довольно сложный вариант, когда игра вышла примерно в ноль. Бывают случаи, когда, исправив что-то, можно выйти на неплохой заработок. Тогда принимается решение выделить дополнительные ресурсы, если они, конечно, есть. Игра временно становится убыточной, чтобы после устранения проблем начать зарабатывать. Некоторые компании, чтобы выйти в плюс, сокращают команду или переводят ее на другие проекты. Для гейм-дизайнеров подобный вариант может стать иногда даже хуже провального. Такие игры обычно не признаются неудачными сразу, и появляется желание аврально, за счет кранчей^[80] все поправить. Если это помогло, то

замечательно, но вытащить проект удастся далеко не всегда, а деньги и время уже потрачены, команда демотивирована.

После запуска над рядом проектов продолжается активная работа: это могут быть DLC, добавление новых механик, контента, ивентов, монетизационных акций и пр. Поддержка игры после запуска (оперирование) – большая тема, мы рассмотрим ее в отдельной книге. Если проект «выстрелил» и хорошо зарабатывает, поддержание более чем оправданно; такие игры могут жить десятилетиями, привлекая новых поклонников и новых сотрудников, избавляясь от мельчайших багов, стремясь к совершенству. Создатели подобных игр могут рассчитывать и на яхту, и на кругосветные путешествия, и на коллекцию спортивных автомобилей.

К сожалению, серьезного успеха добиваются единицы: все-таки в большинстве своем игра считается успешной, уже зарабатывая хоть что-то. Команды разработчиков таких игр редко вырастают после релиза – скорее, наоборот, их ждет постепенное сокращение, хотя проект может жить еще несколько лет. Просто наступает момент, когда вкладывать ресурсы в развитие игры становится невыгодно; лучше использовать их для создания чего-то нового.

Хорошая практика – анализ проделанной работы: что мы планировали, что в итоге получилось, а что нет, и какие уроки мы можем из этих результатов извлечь. Не всегда этот подход оформляется официально, можно просто собраться в кафе и обсудить запущенный проект. Но крупные компании нередко составляют отдельный документ и даже выкладывают его в общий доступ, так что при желании можно ознакомиться с примерами такого анализа. Этот итоговый документ называют «постмортемом».

Люди растут профессионально, только понимая свои ошибки и свои успехи. И особенно это важно для людей, чьи решения действительно оказывали влияние на судьбу игры. Уверены, что наши читатели – из их числа.

Друзья, мы подошли к концу нашего... учебника? Пособия? Конспекта? Сложно дать определение нашей книге, ведь ясно, что создание игры – настолько комплексная вещь, что было бы наивно пытаться уместить всю необходимую информацию в пару сотен страниц текста. Мы видели свою задачу в том, чтобы обозначить ключевые точки развития игрового проекта, дать практические советы, но главное – помочь сформировать общее гейм-дизайнерское видение, как же делают разные игры.

Гейм-дизайн – не наука в полном смысле этого слова. Это практика и развитие, постоянный поиск нового и уникального. Работа в игровой индустрии – это вызовы, возможность воплотить свой творческий потенциал и сделать счастливее других людей. Не бойтесь трудностей и дерзайте! Придумывайте, тестируйте, ошибайтесь, но делайте свои игры! Поверьте, оно того стоит.

От авторов

КОНСТАНТИН САХНОВ

(Игровой продюсер, 15 лет в игровой индустрии.)

Не ошибается тот, кто ничего не делает.

Впервые я услышал эту истину в игре *Warcraft*, где шаман Нер'Зул открыл темный портал, чтобы спасти свой народ, пожертвовав целой планетой. Все это как нельзя кстати применимо к игровой индустрии.

Геймдев – это место тяжелой и упорной работы. Здесь нужно вкладывать много времени и усилий, ошибаться вновь и вновь, чтобы получить бесценный опыт.

Я начал свой путь в геймдев в 2005-м. Я приносил жертвы и получал награды. Но, достигнув желанной цели, мы так часто забываем, что успех не вечен. Наши достижения со временем теряют ценность. И что бы вы ни сделали в прошлом, вам нужны новые свершения. Новые игры, вселяющие восторг в сердца людей.

Иногда я задумываюсь, сколько времени у нас есть на активную работу. Тридцать, может, сорок лет? Сколько игр вы успеете сделать, сколько из них будут успешны? Что мы подарим нашему игроку, какой вклад внесем в индустрию?

С детства, играя в игры, я проектировал на бумажках свои уровни, думал, как бы я усовершенствовал геймплей. Я выбрал эту индустрию, потому что считал, что делать игры – это так же весело, как в них играть. Разумеется, это не так. Но это тоже очень интересно.

Геймдев – одна из немногих сфер, где мы можем заниматься творчеством, самореализацией. Но важно видеть картину целиком. Это еще и большой бизнес. Многим не нравится этот факт. Люди считают, что деньги губят креатив. На самом деле правда, как всегда, глубже. Если вы не будете рассматривать ваши продукты с точки

зрения бизнеса, на что будет жить ваша команда? Офис, зарплаты, маркетинг – игры должны прежде всего окупаться и быть прибыльным бизнесом. Если этого нет, любое творчество, к сожалению, закончится очень быстро.

Вы попали в уникальную сферу! Здесь самореализация, удовольствие от работы и большие деньги сосуществуют вместе. И лишь сбалансированный подход к работе и продуктам позволит получить от жизни все.

Пробуйте! Потому что не ошибается тот, кто ничего не делает.

ВЯЧЕСЛАВ УТОЧКИН

(Более 10 лет в игровой индустрии. Директор образовательных программ по игровой индустрии Центра развития компетенций в бизнес-информатике Высшей школы бизнеса НИУ ВШЭ. Со-основатель и генеральный продюсер международного издателя и разработчика игр Geeky House.)

Работа в игровой индустрии – это не только профессия, приносящая зарплату, но и настоящее удовольствие от вложенного труда. Я связал свою жизнь с геймдевом в 25 лет, а до этого успел поработать и в маркетинге, и в недвижимости, и в банке. Когда ты проводишь рабочее время с удовольствием, возникает совершенно другое ощущение от жизни в целом. Хочется с утра идти на работу, от работы горят глаза, а видимый результат того, что ты делаешь, влияет на тысячи, сотни тысяч, а иногда даже и на миллионы игроков по всему миру!

Но первое время было очень сложно вникнуть в новую для себя сферу, особенно когда ты уже состоявшийся специалист в совершенно другой области, а к играм имеешь отношение только как геймер. Тогда еще не было геймдев-образования, почти не встречались книги о разработке игр на русском языке и было мало профильных интернет-ресурсов. Именно тогда мне в душу запала

мечта выстроить для тех, кто так же, как я, хочет создавать собственные игровые миры, удобную систему погружения в эту индустрию. Чтобы было кому помочь, направить, дать нужные инструменты и подсказки на этом пути.

Поэтому, когда я и мой друг Костя Сахнов, с которым мы и начинали вместе погружение в мир разработки игр, сами стали опытными разработчиками, мы приняли активное участие в создании геймдев-образования в нашей стране. В частности, в запуске и развитии программы профессиональной переподготовки «Менеджмент игровых проектов» в Вышке, которая уже помогла сотням людей перейти в геймдев из другой сферы. Для многих из них, как и для меня, создание собственных игр приносит теперь больше удовольствия, чем просто играть в крутые игры.

В этой книге мы собрали выжимку из всех тех знаний, которые получили за десять с лишним лет работы в игровой индустрии, которые систематизировали в рамках образовательных программ по геймдеву и привели к финальной форме. Это стало возможным благодаря помощи десятков наших коллег и выпускников, которые читали книгу на этапе ее бета-теста, комментировали и вносили в нее частичку своего опыта и своей души. Мы все подходили к этому делу так, как если бы писали сами для себя, если бы мы решили с нуля пойти в новую сферу.

Мы очень надеемся, что вам, нашим читателям, книга смогла дать вектор развития и помогла сделать первый шаг на пути к созданию крутых игр, к творческой и профессиональной самореализации!

Благодарности

В современном мире все реже можно встретить продукты, созданные одним человеком. За YouTube-блогером зачастую стоит команда монтажеров, редакторов, сценаристов, осветителей, маркетологов и других работников. Так и за нашей книгой стоит команда, приложившая немало усилий для того, чтобы знания авторов были тщательно изложены на бумаге. Опыт каждого соавтора упакован емкими фразами, их работа скоординирована для рождения единой цепи повествования, а профессиональный сленг адаптирован для читателя с разным уровнем подготовки. А что стоит одна только работа с экспертами! Их жизнь полна конференций, лекций, кропотливой работы и крупных статей. Лавировать в их расписании и достигать результата – отдельный большой талант. Эта книга не появилась бы в том виде, в котором вы сейчас держите ее в руках, слушаете или читаете с экрана, если бы не огромная работа нашего редактора Анастасии Москвиной. К слову, также выпускницы программы «Менеджмент игровых проектов» в Центре развития компетенций в бизнес-информатике Высшей школы бизнеса НИУ ВШЭ, где преподают авторы книги. Коллектив авторов считает своим долгом выразить ей благодарность в том числе от лица читателей, которые теперь имеют возможность узнать об игровой индустрии чуточку больше.

Огромное спасибо за ценные комментарии и рекомендации:

Вовку Михаилу

Табакову Дмитрию

Доброштану Олегу

Голубкину Сергею

Чекмаеву Сергею

Зыкову Сергею

Лукьяненко Славе

Gray Richard

**Зезюлиной Елене
Кулешову Степану
Крохину Илье
Луковатому Вадиму
Клейменову Вячеславу
Михно Евгению
Ершову Михаилу
Величко Вере
Красильникову Владимиру
Макушенко Ольге
Кириченко Карине
Полякову Григорию
Светлову Дмитрию
Исаенкову Дмитрию
Коломиец Наталье**

* * *

ЛУЧШИЕ КНИГИ О БИЗНЕСЕ С ЛОГОТИПОМ ВАШЕЙ КОМПАНИИ? ЛЕГКО!

Удивить своих клиентов, бизнес-партнеров, сделать памятный подарок сотрудникам и рассказать о своей компании читателям бизнес-литературы? Приглашаем стать партнерами выпуска актуальных и популярных книг. О вашей компании узнает наиболее активная аудитория.

ПАРТНЕРСКИЕ ОПЦИИ:

- Специальный тираж уже существующих книг с логотипом вашей компании.
- Размещение логотипа на супер-обложке для малых тиражей (от 30 штук).
- Поддержка выхода новинки, которая ранее не была доступна читателям (50 книг в подарок).

ПАРТНЕРСКИЕ ВОЗМОЖНОСТИ:

- Рекламная полоса о вашей компании внутри книги.
- Вступительное слово в книге от первых лиц компании-партнера.
- Обращение первых лиц на суперобложке.
- Отзыв на обороте обложки вложение информационных материалов о вашей компании (закладки, листовки, мини-буклеты).



У вас есть возможность обсудить свои пожелания с менеджерами корпоративных продаж. Как?

Звоните:

+7 495 411 68 59, доб. 2261

Заходите на сайт:

eksmo.ru/b2b



ХОЧУ В ГЕЙМДЕВ!



ОСНОВЫ ИГРОВОЙ РАЗРАБОТКИ ДЛЯ НАЧИНАЮЩИХ

ВЯЧЕСЛАВ УТОЧКИН
КОНСТАНТИН САХНОВ

 **БОМБОРА**
ИЗДАТЕЛЬСТВО

Примечания

1

. *Инди-игра* – компьютерная игра, созданная отдельным разработчиком или небольшим коллективом без финансовой поддержки издателя компьютерных игр. – Здесь и далее примечания авторов.

[Вернуться](#)

2

. *Гик* (от англ. *geek*) – человек, чрезвычайно увлеченный чем-либо; фанат. Изначально гиками именовали людей, увлеченных высокими технологиями (обычно компьютерами и гаджетами).

[Вернуться](#)

3

. *Хилер, хил* (от англ. *heal* – «исцелять») – целитель.

[Вернуться](#)

4

. *Паблик* – группа или сообщество в социальных сетях.

[Вернуться](#)

5

. *Матч-3, или три-в-ряд (от англ. match-three), – жанр компьютерных игр, где игровой мир состоит из таблицы или сетки элементов. Ими нужно манипулировать таким образом, чтобы совпали заданные комбинации, после чего собранные элементы исчезают.*

[Вернуться](#)

6

. *Ачиверы (от англ. achieve – «достигать», «добиваться»).*

[Вернуться](#)

7

. *Киллеры (от англ. kill – «убивать»).*

[Вернуться](#)

8

. *Нейромедиаторы – биологически активные вещества, являющиеся химическими передатчиками в организме.*

[Вернуться](#)

9

. *UX (от англ. user experience – «опыт пользователя», «опыт взаимодействия»). В гейм-дизайне есть одноименное направление, отвечающее за то, какой опыт/впечатления получает пользователь от взаимодействия с интерфейсом игры.*

[Вернуться](#)

10

. Лутбокс – виртуальный предмет в компьютерных играх, при использовании которого игрок получает случайные виртуальные предметы различной ценности и назначения.

[Вернуться](#)

11

. Хидео Кодзима – японский гейм-дизайнер, сценарист, продюсер и руководитель разработки компьютерных игр. Известен своеобразным, авторским подходом к созданию игр. Создатель серии *Metal Gear*, *Death Stranding* и других культовых игр.

[Вернуться](#)

12

. Лор (от англ. *lore* – «знания») – совокупность истории и сюжета компьютерной игры. Это та информация о мире («вселенной») игры, которую можно получить, играя в нее, а также изучая дополнительные материалы (книги, комиксы, фильмы и пр.).

[Вернуться](#)

13

. Сеттинг (от англ. *setting* – «помещение», «установка», «обстановка») – среда, в которой происходит действие; место, время и условия действия. Сеттинг в художественных произведениях (книгах, фильмах, настольных и компьютерных играх) – это описание мира, в котором происходит действие произведения.

[Вернуться](#)

14

. Хард-скиллы (от англ. *hard skills* – «твердые навыки») – прикладные навыки в определенной профессии. Им можно научить, их можно продемонстрировать, их указывают в должностных инструкциях.

[Вернуться](#)

15

. Софт-скиллы (от англ. *soft skills* – «мягкие навыки») – навыки, связанные не с конкретным видом деятельности, а с коммуникациями для эффективного взаимодействия.

[Вернуться](#)

16

. Компьютерная ролевая игра (от англ. *Computer Role-Playing Game*, обозначается аббревиатурой *CRPG* или *RPG*) – жанр компьютерных ролевых игр, основанный на элементах игрового процесса традиционных настольных ролевых игр. В ролевой игре игрок управляет одним или несколькими персонажами, каждый из которых обладает набором численных характеристик, списком способностей и умений; примерами таких характеристик могут быть очки здоровья (от англ. *hit points* или *health points* (HP)), показатели силы, ловкости, интеллекта, защиты, уклонения, уровень развития того или иного навыка и т. п. Обычно создатели *RPG* полагаются на продуманный сюжет и игровой мир.

[Вернуться](#)

17

. Junior (от англ. *junior* – «младший») – специалист, только начинающий карьеру.

[Вернуться](#)

18

. ТЗ – сокращенно от «техническое задание».

[Вернуться](#)

19

. Контент – информационно значимое наполнение чего-либо.

[Вернуться](#)

20

. Часто в названии эта приписка опускается, и должность звучит просто как Game Designer.

[Вернуться](#)

21

. Абилка (от англ. ability – «способность», «умение»).

[Вернуться](#)

22

. Senior (от англ. senior – «старший») – специалист-профессионал, обычно один из лучших сотрудников в компании.

[Вернуться](#)

23

. Фича (от англ. *feature* – «особенность», «необычное свойство», «фишка») – сленговое обозначение каких-либо особенностей. В игровой индустрии так называют обособленную часть игрового функционала, определенную игровую систему (система прицеливания, крафт оружия и т. д.).

[Вернуться](#)

24

. Lead (от англ. *lead* – «лидер», «руководитель») – ведущий специалист на проекте. В русском сленге часто называют «лид».

[Вернуться](#)

25

. Task-трекер (от англ. *task tracker*) – специализированный сайт или программа, позволяющая заводить задачи, смотреть статусы их выполнения, отвечать на вопросы и т. д.

[Вернуться](#)

26

. Игровой ассет (от англ. *game asset*) – цифровой объект, часть игрового контента, обладающая некими свойствами (текстуры, 3D-модели, файлы анимации и др.).

[Вернуться](#)

27

. Баг (жарг.) – программная ошибка.

[Вернуться](#)

28

. Плейтест – тестирование игры с целью сбора обратной связи.

[Вернуться](#)

29

. Дорожная карта (от англ. roadmap) – это визуальное представление стратегии реализации проекта или очень высокоуровневого плана с основными вехами.

[Вернуться](#)

30

. Под закупкой трафика имеется в виду покупка рекламы игры в социальных сетях и других ресурсах. Трафик – это пользователи, которые играют в наши игры, наша аудитория. Платный трафик – это реклама, которую разработчики игры покупают на различных ресурсах с целью привлечь новых пользователей.

[Вернуться](#)

31

. USP (от англ. Unique Selling Points или Unique Selling Proposition) – уникальные особенности игры.

[Вернуться](#)

32

. Core-геймплей (от англ. gameplay loop, core gameplay loop) – принцип, согласно которому гейм-дизайнеры задают главный элемент игровой механики, который определяет фундаментальный

опыт игрока. Один игровой цикл представляет собой действие игрока, результат этого действия в игровом мире, реакцию игрока на результат и запрос игры на повторение нового действия.

[Вернуться](#)

33

. DLC (от англ. downloadable content) – форма распространения дополнительного игрового контента.

[Вернуться](#)

34

. SWOT-анализ – метод стратегического планирования, заключающийся в выявлении факторов внутренней и внешней среды и разделении их на четыре категории:

- Strengths – «сильные стороны»;*
- Weaknesses – «слабые стороны»;*
- Opportunities – «возможности»;*
- Threats – «угрозы».*

[Вернуться](#)

35

. Гейм-дизайн-документ (сокр. ГДД или диздок) – это детальное описание ключевых аспектов разрабатываемой игры. Диздок – это тип документации, и на крупных проектах он может делиться на множество гейм-дизайн-документов: например, ГДД по каждой отдельной фиче внутри одной игры.

[Вернуться](#)

36

. *Waterfall, «Каскадная модель» (от англ. waterfall model – «модель «Водопад»)* – методика управления проектами, которая подразумевает последовательный переход с одного этапа на другой без пропусков и возвратов на предыдущие стадии.

[Вернуться](#)

37

. *Спринт* – это короткий период времени, в течение которого команда *Scrum* работает над выполнением заданного объема работы.

[Вернуться](#)

38

. *Бэклог (от англ. product backlog)* – это упорядоченный набор элементов, очередь задач, перечень всех функций продукта.

[Вернуться](#)

39

. *Майлстоун* – контрольная точка, важный момент, этап. Исторически майлстоуны – это каменные плиты с указанием расстояний до населенных пунктов, которые расставлялись вдоль дорог и служили ориентиром путешественникам.

[Вернуться](#)

40

. *Фичекрип (от англ. feature creep)* – насаждение возможностей, разрастание фичей или одной конкретной фичи.

[Вернуться](#)

41

. *Фичекат* (от англ. *feature cut*) – удаление или урезание функционала лишних фичей.

[Вернуться](#)

42

. *Блупринты* (от англ. *blueprints*) – это система визуального программирования в игровом движке *Unreal Engine*.

[Вернуться](#)

43

. *GameMaker* – простой игровой движок, не требующий навыков программирования.

[Вернуться](#)

44

. *Аутлайн* – контур вокруг игровой сущности (предмета или персонажа).

[Вернуться](#)

45

. *AAA* – неформальный термин для обозначения самых высококлассных, высокобюджетных игр, рассчитанных на массовую аудиторию. Аналогично «блокбастеру» в киноиндустрии.

[Вернуться](#)

46

. Чит-код (от англ. cheat code, cheat – «жульничество», «обман») – стороннее ПО для получения нечестного преимущества; иногда отладочный код – код, который может быть введен в программу, чтобы изменить ход ее работы.

[Вернуться](#)

47

. RTX – это технология, создающая реалистичное освещение, тени и отражения на объектах. Теоретически уровень реализма с трассировкой лучей должен быть на порядок выше, чем при традиционных способах рендеринга.

[Вернуться](#)

48

. Гейм-джем (от англ. game jam) – сбор разработчиков игр с целью разработки одной или нескольких игр за ограниченный промежуток времени.

[Вернуться](#)

49

. Высокополигональный – состоящий из большого количества полигонов. Полигоны – плоскости, образованные из вершин, из которых состоят 3D-модели. Чем их больше, тем больше нагрузка на рендер.

[Вернуться](#)

50

. Гиперказуальные игры – это простейшие игры с затягивающим геймплеем и минималистичным артом, в которые обычно играют только с помощью «тапов» по экрану. У них простые и понятные цели – набрать больше очков, зачистить уровни и так далее.

[Вернуться](#)

51

. Нон-геймеры – пользователи игр, не относящие себя к игрокам или играющие впервые.

[Вернуться](#)

52

. Органический трафик – пользователи, пришедшие с помощью запросов в поисковых системах или по совету знакомых. Органическим его называют, так как он естественный, бесплатный.

[Вернуться](#)

53

. Пайплайн – цепочка процессов разработки продукта.

[Вернуться](#)

54

. Фрилансер – это вольнонаемный; человек, который сам берет отдельные заказы. Он не работает на кого-то постоянно, и у него нет фирмы, оказывающей постоянную услугу, но он выполняет работы для разных заказчиков.

[Вернуться](#)

55

. Гринд (от англ. *grind, grinding* – «молоть») – в компьютерных играх повторяющиеся и связанные с небольшим риском действия игроков, направленные на получение внутриигровой выгоды.

[Вернуться](#)

56

. Пропсы в геймдеве – это своего рода бутафория, позволяющая игроку ощутить атмосферу и оценить обстановку в окружающем мире.

[Вернуться](#)

57

. NPC (от англ. *Non-Player Character* – «неигровой персонаж») – персонаж в играх, которым управляет не игрок, а компьютер или мастер.

[Вернуться](#)

58

. Спавн (от англ. *respawn* – «возрождение») – определенное место, в котором появляется какой-либо предмет или NPC.

[Вернуться](#)

59

. Лут – любые предметы, ресурсы или деньги, выпадающие из NPC после их убийства, которые игрок может забрать себе. Другими словами, добыча или трофеи.

[Вернуться](#)

60

. Реиграбельность – качественная характеристика игры, которая определяет степень того, насколько игроки хотят сыграть в рассматриваемую игру еще раз, даже если они ее уже «прошли», или после того, как уделили большое количество времени и достигли некоторого уровня мастерства.

[Вернуться](#)

61

. QA (от англ. quality assurance – «обеспечение качества») – тестировщик.

[Вернуться](#)

62

. Винрейт – процент побед среди всех сыгранных игр.

[Вернуться](#)

63

. PvE (от англ. player versus environment – «игрок против окружения»), или PvM (от англ. player versus monster – «игрок против монстра»), – термин, используемый в онлайн-играх для обозначения сражений с управляемыми компьютером врагами в противовес термину PvP (от англ. player versus player – «игрок против игрока»).

Преимущественно встречается в жанрах MMORPG, MUD и других онлайн-вариациях компьютерных ролевых игр.

[Вернуться](#)

64

. Система рейтингов Эло, коэффициент Эло – метод расчета относительной силы игроков в играх, в которых участвуют двое игроков (например, сего, го или шахматы).

[Вернуться](#)

65

. Мета (от англ. meta gameplay, metagame), или метаигра, – совокупность механик или других игровых активностей вне базового игрового процесса (core gameplay). Вместе они составляют саму игру со всем многообразием игровых механик. В ККИ и настольных играх под этим термином также понимают процесс, сопровождающий саму игру, в частности, коллекционирование карт, подбор миниатюр и других элементов компонентной базы игры, участвующих в основном игровом процессе. Здесь и далее мы будем иметь в виду мету в первом ее значении, если иное не указано отдельно.

[Вернуться](#)

66

. Рогалики (от англ. roguelike) – жанр компьютерных игр, поджанр компьютерных ролевых игр. Характерными особенностями классического roguelike являются генерируемые случайным образом уровни, пошаговость и необратимость смерти персонажа.

[Вернуться](#)

67

. Донат (от англ. donate – «жертвовать», «дарить») – в играх обычно обозначает оплату игроком реальными деньгами каких-либо дополнительных бонусов, уникальных предметов и прочих благ.

[Вернуться](#)

68

. Paywall – система, при которой часть контента в игре доступна лишь за определенную плату.

[Вернуться](#)

69

. Pay-to-win – это система монетизации, при которой пользователям предлагается приобретение игровых преимуществ за реальные деньги. Размер полученного преимущества пропорционален количеству потраченных денег, а возможность вкладывания денежных средств ничем не ограничена.

[Вернуться](#)

70

. HTML (от англ. HyperText Markup Language – «язык гипертекстовой разметки») – стандартизированный язык разметки документов в Интернете. Большинство веб-страниц содержат описание на этом языке.

[Вернуться](#)

71

. Транзакции в СУБД – атомарные действия над БД, переводящие ее из одного целостного состояния в другое целостное состояние. Другими словами, транзакция – это последовательность операций, которые должны быть или все выполнены, или все не выполнены (все или ничего).

[Вернуться](#)

72

. Figma – кросс-платформенный онлайн-сервис для дизайнеров.

[Вернуться](#)

73

. Эффект «зловещей долины» (от англ. uncanny valley) – гипотеза, по которой робот или другой объект, выглядящий или действующий примерно как человек (но не точно так, как настоящий), вызывает неприязнь и отвращение у людей-наблюдателей.

[Вернуться](#)

74

. CGI (от англ. Computer-Generated Imagery – «изображения, сгенерированные компьютером») – неподвижные и движущиеся изображения, сгенерированные при помощи трехмерной компьютерной графики и используемые в изобразительном искусстве, печати, кинематографических спецэффектах, на телевидении и в симуляторах.

[Вернуться](#)

75

. Рекомендуем ознакомиться с трудом В. Я. Проппа «Исторические корни волшебной сказки».

[Вернуться](#)

76

. DRM (от англ. Digital Rights Management) – технические средства защиты авторских прав.

[Вернуться](#)

77

. Тестовое покрытие (от англ. test coverage) – это одна из метрик оценки качества тестирования программного обеспечения. Представляет собой величину, выраженную в процентах, которая отображает степень плотности покрытия тестами функциональных требований или исполняемого кода.

[Вернуться](#)

78

. ЗБТ – сокращенно от «закрытое бета-тестирование».

[Вернуться](#)

79

. Таргетинг (от англ. target – «цель») – рекламный механизм, позволяющий выделить из всей имеющейся аудитории только ту часть, которая удовлетворяет заданным критериям (т. е. целевую аудиторию).

[Вернуться](#)

. Кранч – аврал: рабочая ситуация, в которой сотрудникам приходится работать в день больше часов, чем они должны по трудовому договору, чтобы выполнить срочные задачи в срок.

[Вернуться](#)