13

Зависимости

После создания заповедника недавно обученные смотрители парка сразу решили внести некоторые изменения. Они думали, что лосей было недостаточно, поэтому запустили программу по их вскармливанию.

Популяция лосей значительно выросла. Вскоре огромные стада лосей начали съедать осины и ивы. Это привело к исчезновению бобров, так как у них не осталось достаточно древесины для плотин. Без плотин парк начал пересыхать каждое лето. Для рыбы не осталось водоемов, и вскоре озера опустели. С исчезновением рыбы упала популяция гризли, так как они выживали благодаря пойманной из рек рыбе. С исчезновением гризли и появлением такого количества лосей, на которых можно охотиться, выросла популяция волков. Рост популяции оленей вскоре прекратился из-за волков и выпаса лосей.

И изменения продолжались, распространяясь на всю экосистему...

Геймдизайн может иметь сотни механик, вымышленных элементов и подсистем. Даже в течение нескольких минут после создания идеи игры у дизайнера может быть 20 различных идей для задач, систем и интерфейсов, которые можно добавить. Имея такое количество идей, как мы узнаем, над чем работать в первую очередь? Начнем ли мы с самого уникального? Самого основного? Самого легкого? Самого технологичного? Самого рискованного?

Ключ к ответу на этот вопрос заключается в понимании зависимостей.

ЗАВИСИМОСТЬ — это такое отношение между двумя частями проекта, когда изменения в одной части вызывают изменения в другой.

Представьте, что кто-то просит вас покрасить 10 домов. В этой задаче нет зависимости. Неважно, в каком порядке вы будете их красить, потому что это не влияет на то, как вы будете красить другие дома.

В геймдизайне все не так. Различные части дизайна часто взаимозависимы. Внешний вид уровня зависит от оформления этого уровня. Оформление зависит от инструментов

игрока. Инструменты игрока зависят от базового интерфейса. Если какой-либо элемент изменяется, то изменяется и каждый зависимый от него элемент.

Понимание зависимостей помогает нам снизить риск того, что придется переделывать законченную работу из-за изменений в каком-то месте, от которого она зависела. Например, представьте, что мы потратили время на анимацию персонажа, который бегает со скоростью 5 километров в час во всех направлениях. Если позже мы решим, что он должен двигаться со скоростью 7 километров в час, всю эту анимацию придется переделывать.

Анимация зависит от дизайна системы движений персонажа; изменение в системе движений повлияет на анимацию и испортит хорошую работу. Если бы мы лучше понимали зависимости, мы могли бы сначала укрепить механику движения (в «сером ящике»), а потом делать анимацию.

Стек зависимостей

Чтобы понять зависимости в дизайне, дизайнеры могут построить стек зависимостей.

Чтобы создать стек зависимостей, мы начинаем с геймдизайна. Проект может быть представлен в виде плана на бумаге в начале разработки или он может быть частично реализован и протестирован. Мы разбиваем игру на отдельные элементы — механику, элементы управления, интерфейсы и подсистемы. Затем определяем ключевые зависимости между этими элементами. Наконец, мы рисуем график, иллюстрирующий все взаимозависимости. Это и есть стек зависимостей.

СТЕК ЗАВИСИМОСТЕЙ — это простой метод анализа, который определяет ключевые зависимости между элементами проекта. Так мы можем понять, над чем нужно работать сейчас, а что можно оставить на потом.

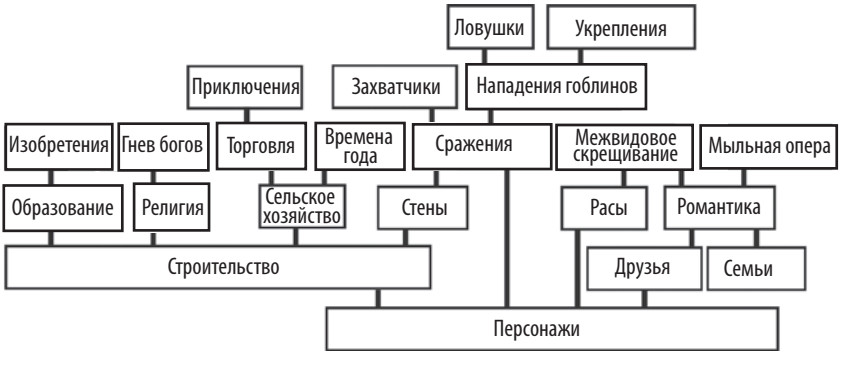
Давайте разберем пример. Представьте, что мы делаем Fantasy Castle, легкую игру о строительстве замка в фэнтези-мире. Разработка Fantasy Castle только началась, поэтому команда дизайнеров переполнена идеями, но им не хватает протестированных и проверенных проектных решений. Они написали длинный дизайн-документ. Каждая из 22 подсистем подробно описана на бумаге. Далее следует их краткое изложение в произвольном порядке:

- **Персонажи.** Могут существовать и перемещаться в окружающей среде.
- **Семьи.** Персонажи могут вступать в семейные отношения.
- **Расы.** Замок может быть заполнен большой группой людей, эльфов, гномов и других фэнтезийных рас.
- **Межвидовое скрещивание.** Различные виды могут скрещиваться, чтобы создать гибриды с общими характеристиками своих родителей.
- **Нападения гоблинов.** Гоблины могут устраивать периодические облавы, чтобы проверять оборону замка.
- **Сельское хозяйство.** Сельское хозяйство и продовольственная система дают населению пропитание.
- **Торговля.** Персонажи могут торговать с соседними замками особыми или редкими товарами.
- **Образование.** Персонажи могут получить образование.
- **Изобретения.** Образованные персонажи могут изобретать новые машины для замка.
- **Религия.** Персонажи могут строить храмы и проводить там религиозные обряды, развивать отношения с конкретными божествами и получать дары от них.
- **Гнев богов.** Другие божества разозлятся на вас, если вы будете неуважительно относиться к ним или поклоняться их врагам.
- **Друзья.** Персонажи могут иметь платонические отношения.
- **Романтика.** Персонажи могут иметь романтические отношения.
- **Строительство.** Персонажи могут что-то строить.
- **Стены.** Персонажи могут строить стены, чтобы остановить врагов.
- **Укрепления.** Стены можно укреплять и утолщать, чтобы сдерживать гоблинов.

- **Ловушки.** Персонажи могут устанавливать ловушки или автоматические защиты.
- **Сражения.** Персонажи могут бороться, чтобы защитить замок.
- **Захватчики.** Можно организовывать группы захвата, чтобы исследовать близлежащие подземелья и получать добычу.
- **Приключения.** Можно обслуживать путешествующих искателей приключений, обеспечивая трактиры и лавочников в обмен на трофеи из древних склепов.
- **Времена года.** Полный сезонный цикл влияет на сельское хозяйство, строительство и другие виды деятельности.
- **Мыльная опера.** Среди жителей замка разыгрываются адюльтер и прочие романтические драмы.

Игра может включать в себя все эти элементы, но как выбрать, на чем сосредоточиться в первую очередь? Начинать ли с основ мира и создавать времена года? Начинать с персонажей, друзей и семьи? Возможно, следует начать с войны, выстроив боевую систему для сражений между жителями замка и гоблинами? Начать с уникальных элементов игры, таких как искатели приключений, или построить базу, используя элементы, которые были сделаны ранее в других играх, такие как стены и бои? Стек зависимостей помогает нам принять решение.

Ниже представлен мой стек зависимостей для игры Fantasy Castle:



Нет смысла в ловушках и оборонительных системах, если нет набегов гоблинов. Набеги гоблинов не нужны, если нет боя. Бой не имеет особого значения при отсутствии стен и не может работать без персонажей. Для стен нужны системы строительства, которые требуют, чтобы персонажи работали. Каждый элемент зависит от нижестоящих элементов.

Прежде чем мы продолжим, позвольте мне прояснить понятие зависимости.

Помните, что блоки стека — это не просто слова, которые вы видите, — каждое слово представляет собой подробный дизайн нескольких страниц. Части этих дизайнов взаимосвязаны. Например, дизайн строительства описывает каждую кнопку, выделение и опцию интерфейса, который игроки используют для конструирования. Аналогичным образом дизайн фермерства подробно описывает размещение, управление и удаление ферм. При этом дизайн фермерства предполагает, что дизайн строительства будет правильным. Что произойдет, если изменятся детали в интерфейсе строительства, если кнопки поменяются местами или будут заменены на управление мышью? Строительство все еще остается в игре, но оно изменилось, и теперь все детали фермерства необходимо изменить соответствующим образом. Вот почему фермерство зависит от строительства.

ЗАВИСИМОСТЬ не означает, что основополагающий элемент не должен влиять на зависимый элемент. Она предполагает, что изменения в дизайне базового элемента приведут к изменениям в зависимом элементе.

Существует гораздо больше зависимостей подобного типа, чем записано в стеке. Например, я разместил стек «стены» над стеком «строительство». Но возможно, что в процессе разработки дизайнеры могут обнаружить, что им нужно изменить интерфейс строительства, чтобы упростить строительство стен. Таким образом, стек «стены» зависит от стека «строительство», а стек «строительство» зависит от стека «стены» — циклическая зависимость. Эти нити зависимостей пронизывают Fantasy Castle насквозь. Возможно, придется изменить стек «стен», чтобы их можно было легче размещать вокруг ферм. На систему изобретений может повлиять стек «дружеские взаимоотношения», если друзья могут настраивать друг друга на создание изобретений.

В каком-то смысле любой элемент может повлиять на все остальное, поэтому все взаимозависимо.

Но некоторые из этих зависимостей сильнее, чем другие. Значительные изменения в стеке «стены» могут в некоторых случаях повлиять на стек «строительство». Но изменения в стеке «строительство» почти наверняка затронут стек «стены». Стек зависимостей намеренно игнорирует самые слабые зависимости, чтобы мы могли сосредоточиться на наиболее важных и потенциально опасных из них. Нахождение этого фокуса является целью стека.

Стек зависимостей является редуктивным. Он не включает в себя важные части реальности разработки. Но если вы просто человек, столкнувшийся с такой сложной проблемой, как сотня взаимозависимых элементов дизайна, включающих сто тысяч взаимосвязей, рациональная редукция — единственный способ добиться прогресса. Мы должны игнорировать некоторые зависимости, иначе погрузимся в аналитический паралич. Стек зависимостей — это не теоретический вопрос, это инструмент для принятия решений. И эти решения лучше принимать, концентрируясь на наиболее значимых взаимозависимостях. Существует несколько способов построения стека зависимостей с учетом деталей дизайна. Можно представить себе игру по строительству замков, в которой существует строительство, но нет персонажей. Или такую, в которой есть нападение гоблинов, но нет стен. Я создал этот стек на основании придуманных мной деталей для Fantasy Castle, но мне не хватит этой книги для того, чтобы я мог их описать. Если бы дизайн-документы были написаны по-разному, стеки бы тоже были разными, даже если бы названия были одинаковыми.

Каскадная неопределенность

Мы уже видели, что проекты могут не работать так, как мы ожидаем. Дизайнеры могут написать документ о том, как будут работать нападения гоблинов, но они не могут быть уверены, что это будет работать именно так, как предполагалось. Дизайнеры смогут убедиться, будет ли это работать, только когда проведут компилирование и плейтест.