**Лабораторная работа №3**

[**Разработка**](http://zillion.net/ru/blog/354/sozdaniie-profilia-potriebitielia-na-osnovie-karty-empatii) **дизайн-документации**

**Цель работы:** ознакомиться с основными этапами разработки дизайн-документации; получить навыки разработки дизайн-документации игрового приложения.

**Задание:**

1. Провести анализ разработанного в лабораторной работе №2 концепт-документа игры. При необходимости скорректировать его.
2. Изучить план дизайн-документации, определить содержание разделов.
3. Разработать и описать объектные модели игровых сущностей (включая аватаров, если они предусмотрены концептом игры).
4. Разработать и описать функциональную спецификацию (геймплей).
5. Составить описание контента игры.
6. Разработать интерфейс игрового приложения.
7. Описать способы монетизации игры.
8. Разработать и описать механизмы повышения виртуальности игры.

**ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ**

**План дизайн-документации**

В общем случае дизайн документация включает следующие разделы (список может уточняться и изменяться в зависимости от проекта):

1. *Объектная модель.*

Описывает каждую игровую сущность, ее параметры и то, что она может делать.

1. *Функциональная спецификация.*

Описывает геймплей – правила игры, по которым существуют игровые сущности.

1. *Контент игры.*

Включает в себя описание конкретики в игре (разновидности игровых сущностей, их истории, особенности использования и т.п.).

1. *Интерфейс (может быть включен в контент).*

Описание стиля интерфейса, прототипы интерфейса, графы переходов, блок-схемы и т.д.

1. *Монетизация (если предусматривается).*

Описание статей монетизации, как они работают, на кого нацелены, какой вклад в общий доход игры они внесут (в процентном соотношении), скидки, акции и прочее.

1. *Виртуальность (если предусматривается).*

Характеристика, определяющая стремительность распространения контента среди пользователей. Описываются способы повышения виральности игрового приложения.

1. *Техническая спецификация.*

Предназначена для программистов, которые пишут документацию по игре, ее архитектуре, описывают реализацию, кодинг формат, схемы взаимодействия сущностей и т.д.

1. *База знаний.*

Общий ресурс с полезными для проекта материалами (статьи, ссылки, статистические данные, видео, контакты и т.д.).

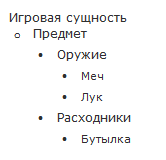
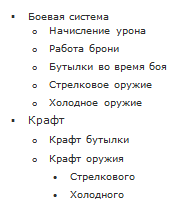
**Объектная модель**

Объектная модель выполняет две важные функции:

1. Справочник игровых сущностей. Его может посмотреть любой член команды без углубления в дизайн-документацию.
2. Структура наследования параметров. Используется при формировании архитектуры будущего проекта.

***Справочник*** позволяет отделить сущность и ее параметры от ее функционала. Это дает возможность быстро, одним взглядом понять, как и из чего она состоит, как хранится в базе, как работает в логике игры и так далее. Такое простое восприятие сущностей позволяет программистам легко понять будущую архитектуру проекта и спроектировать то, как он будет создаваться еще до того, как они начнут воплощать его в коде.

Справочник позволяет сортировать игровые сущности по их типам, в то время как в Функциональной спецификации они сортируются или описываются по функциональности (рис.1).

а) б)

Рисунок 1 – Разница между представлениями игровой сущности   
в объектной и функциональной моделях:

а) объектная модель; б) функциональная модель

Поскольку объектная модель строится в соответствии с принципами объектно-ориентированного программирования, то при описании игровых сущностей можно использовать любые форматы, применимые при описании классов. Тем не менее, наиболее часто используемым является табличный формат (табл. 1). Дополнительно в таблице описывается доступная для сущности функциональность.

Таблица 1 – Табличный формат описания свойств игровой сущности в объектной модели

|  |  |  |
| --- | --- | --- |
| **Описание** | **Тип** | **Имя** |
| <Название свойства> | <Тип в языке программирования> | <ID свойства> |

Доступная функциональность:

<функция 1>

<функция 2>

…

<функция N>

Пример описания сущности и ее функциональности приведен на рис.2.

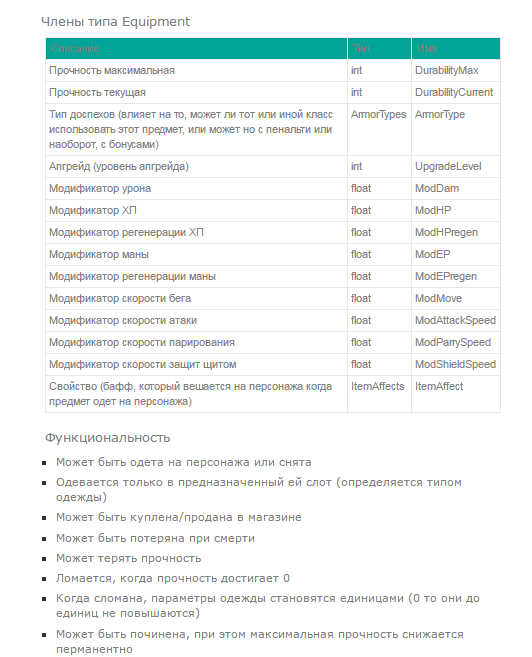


Рисунок 2 – Пример описания сущности и ее функциональности   
в объектной модели

При описании объектной модели рекомендуется использовать механизмы наследования и составлять ***иерархию игровых сущностей.*** Это позволит существенно упростить разработку модели и ее дальнейшую реализацию. Пример фрагмента объектной модели приведен на рис. 3, соответствующие описания сущностей приведены в табл. 2-5.

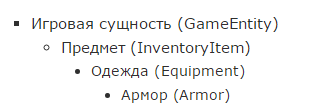


Рисунок 3 – Пример иерархии игровых сущностей

Таблица 2 – Члены типа GameEntity (это верхняя, самая главная сущность в архитектуре проекта и то, как она хранится в БД)

|  |  |  |
| --- | --- | --- |
| **Описание** | **Тип** | **Имя** |
| Рут тайп | Int | RootType |
| Идентификатор сущности, уникальный в группе данного типа сущности | Int | Id |
| Имя | string | GameName |

Таблица 3 – Члены типа InventoryItem

|  |  |  |
| --- | --- | --- |
| **Описание** | **Тип** | **Имя** |
| Вес | int | Volume |
| Стоимость игровой валюты | int | PlayMoneyPrice |
| Стоимость реальной валюты | float | RealMoneyPrice |

Таблица 4 – Члены типа Equipment

|  |  |  |
| --- | --- | --- |
| **Описание** | **Тип** | **Имя** |
| Прочность максимальная | int | DurabilityMax |
| Прочность текущая | int | DurabilityCurrent |
| Тип доспехов | ItemTypes | ItemType |

Таблица 5 – Члены типа Armor

|  |  |  |
| --- | --- | --- |
| **Описание** | **Тип** | **Имя** |
| Тип доспехов (легкий, средний, тяжелый) | ArmorTypes | ArmorType |
| Снижение урона в единицах | Float | DamageReduxF |
| Снижение урона в % | Float | DamageReduxP |

Следует отдельно отметить сущности, которые олицетворяют игроков в игровом пространстве.

Будем рассматривать нечто, олицетворяющее игрока в игровом мире, как ***аватар***. Это то, чем игроки обозначают свое расположение в поле обозрения. В «Монополии» аватары – наперсток, автомобиль, маленькая собачка. В шахматах – король (остальные фигуры здесь для того, чтобы защищать короля). В типичных FPS – солдат или часть оружия на экране. Во многих 3D-видеоиграх игрок видит спину аватара, в то время как он исследует мир.

В некоторых играх аватара в игровом пространстве нет. Вместо этого игрок олицетворяет самого себя. Это справедливо, например, для покера.

Аватар отличен от того, что дизайнеры игр называют «игровыми частицами» или «игровыми объектами», которые представляют из себя материальные предметы, необходимые для того, чтобы в игру можно было играть.

**Функциональная спецификация**

Функциональная спецификация описывает геймплей. Фактически, это правила игры, по которым существуют игровые сущности. Здесь описываются все механики игры.

***Механика игры*** – это правила, которые затрагивают игроков, аватары, игровые частицы (сущности), игровое состояние, поле обозрения и описывают все способы изменения игрового состояния.

***Игровое состояние*** представляет собой набор всей релевантной виртуальной информации, которая может изменяться в течение игры.

***Примеры игровых состояний:***

***Шахматы.***

*Игровое состояние включает в себя набор фигур, их позиции на доске и точную информацию, основанную на уже совершенных ходах (к примеру, кто из игроков имеет право сделать рокировку, какие пешки можно взять на проходе, чья очередь ходить).*

***Покер:***

*Игровое состояние включает в себя: руку каждого игрока и его фишки; размер пота; чья очередь делать ставку; кто скинул карты на текущей руке; какие в колоде остались карты и в каком порядке, и т.д.*

Особенностью игрового состояния является то, что игроки не всегда осведомлены обо всём игровом состоянии в целом. Сегмент игрового состояния, который игрок может видеть, определяется как ***поле обозрения.***

Примеры некоторых общих категорий механик, которые обычно можно найти в играх:

* **Установка.** Всегда должно наличествовать хотя бы одно правило, описывающее то, как игра начинается.
* **Условия победы.** Всегда должно присутствовать хотя бы одно правило, описывающее то, как выиграть в игре. Некоторые игры вроде неограниченных ролевых игр (RPG) не содержат условий победы. Поэтому некоторые дизайнеры не рассматривают таковые в качестве игр. Другие считают, что выполнение особенной задачи — это победа, потому что после этого игрок переключается на следующую задачу.
* **Ход игры.** Кто ходит первым и как? Игра пошаговая или риал-тайм? Если пошаговая, то она начинается с определённого игрока, а потом все ходят по часовой стрелке или игроки выставляют ресурсы на аукцион для борьбы за право ходить первым в каждом раунде, или согласно какому-то другому методу? Если игра риал-тайм и два игрока пытаются делать что-то в одно и то же время, то как это разрешается?
* **Действия игрока.** Иногда определяются как «команды», некоторые из наиболее важных механик, характеризующие то, что игроки могут делать и какой эффект эти действия оказывают на игровое состояние.
* **Определения поля(-ей) обозрения.** Механика точно определяет, какой информацией игрок может обладать в любой отдельный момент времени. Следует отметить, что некоторые механики могут изменять поле обозрения, как пример – частично рассеивающийся при определенных условиях туман войны в RTS.

В общем виде состояние игры может быть определено согласно схеме на рис. 4. Игроки на основе анализа состояния принимают решение о своих действиях, выполняют эти действия, взаимодействуют с другими игроками и правилами, а затем устанавливают игру в новое состояние. Игра может делать часть работы сама независимо от действий игроков.

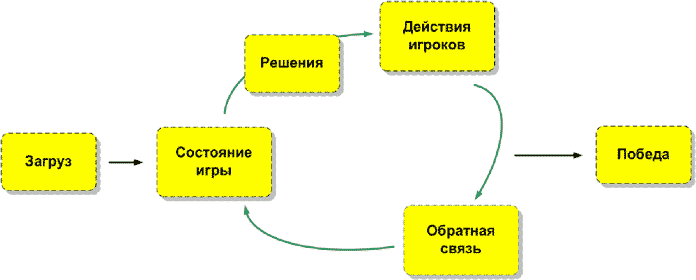


Рисунок 4 – Общая схема игровой механики, как набора факторов, определяющих состояние игры

**Шаг 1: определение состояния.**

Всегда есть некое текущее состояние системы (например, начало хода). Оно определяется на первом такте инициализирующей последовательностью, когда игроки «загружают» игру, раскладывая поле, выбирая фишки, распределяя начальные ресурсы и так далее. Затем оно меняется в зависимости от происходящего в игре.

**Шаг 2: игроки оценивают состояние для принятия решений.**

После обновления состояния игры есть такт, во время которого игроки принимают решения на основе новой информации.

**Шаг 3: игроки выполняют действия.**

На этом шаге игроки меняют состояние игры, делая что-то в соответствии с правилами

**Шаг 4: сама игра даёт обратную связь.**

Например, если бы мы играли в кости, игра сгенерировала бы нам несколько случайных чисел на кубиках для нового состояния.

Следует различать ***Core-механики*** и ***Meta-механики***. Core или основная механика дает инструмент для игры по правилам и алгоритмам в игровом мире. Мета, в свою очередь, создает игру в сознании игрока, запускает мысли, идеи, ожидания и интерес.

Метагейм в широком понимании – всё, что относится к пользователю, а не игре и её персонажам. Рассуждения и мысли игрока, а не его персонажа.

***Примеры Core-механики и Meta-механики.***

***Футбол***

***Core-механика*** *– это непосредственно футбольный матч: бег, пас, навес, прострел, гол. Это футбольные правила, вроде положения вне игры, размеров поля, и количества игровых персонажей на поле.*

***Meta-механикой*** *тут будет выбор тактики на матч: расстановка игроков, выбор состава. Мета-контент – это цифры в характеристиках футболистов, которые запускают процесс анализа и выбора оптимального состава. Здесь одна из мета-механик – это настроение игрока и его усталость, которые влияют на принятие пользователем решения, будет ли футболист заявлен на этот конкретный матч.*

***Покер***

***Core-механика*** *–это стол, карты, ставки, правила.*

***Meta-механика –*** *это стратегия игрока, придержать фишки до последнего стола, не рисковать и играть тайтово, забирать блайнды на префлопе… Это все происходит не конкретно в игре, потому что такой алгоритм. Это происходит потому, что игрок там что-то себе надумал.*

Механика практически всегда является абстракцией, никак не связанной с игровым сеттингом и сюжетом.

**Контент игры**

Включает в себя описание конкретики в игре (разновидности игровых сущностей, их истории, особенности использования и т.п.).

Разница между объектной моделью и контентом – объектная модель описывает шаблон каждой сущности, ***контент описывает экземпляры шаблона*** (аналогия в ООП – «класс»-«экземпляр класса»).

**Интерфейс**

При описании интерфейса минимальный набор требований к UI/UX составляет следующий список:

1. Карта экранов — это диаграмма, где указаны все существующие в игре экраны и всплывающие окна, все возможные переходы между ними на концептуальном уровне.
2. Детальное описание каждого из этих экранов в формате:
   * назначение экрана;
   * список элементов интерфейса и его назначение;
   * рекомендации по механикам элементов интерфейса – описание как именно вы видите работу того или иного переключателя, кнопки, или списка;
   * пожелания по композиции – какие элементы должны получить больше внимания пользователя, а какие отодвинуты на второй план.
3. Дополнительные элементы:
   * примеры реализации;
   * мокапы;
   * возможные тексты, или хотя бы длина в количестве символов;
   * ссылки на приложения, с понравившимися реализациями логики переходов между экранами.

Примеры карт экранов приведены на рис. 5, рис. 6.



Рисунок 5 – Пример неудачной карты экранов

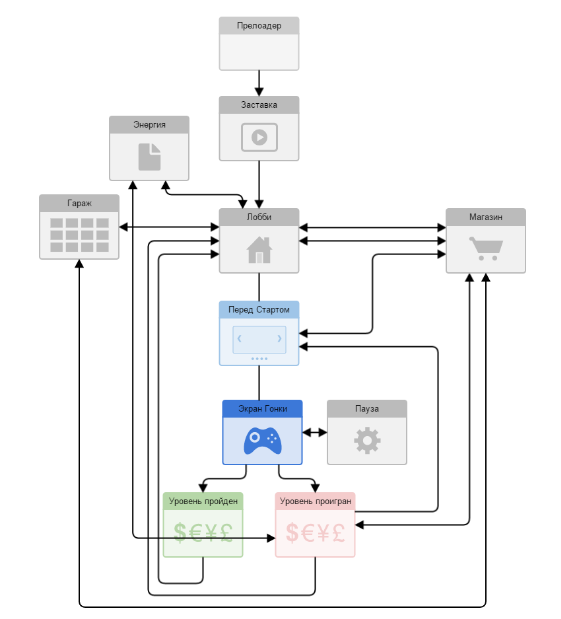


Рисунок 6 – Пример хорошей карты экранов

В первом примере есть три основных минуса:

* мелко и не понятно – дизайнеру крайне не комфортно разбираться в такой структуре, прочувствовать все переходы и сделать хороший UI становится проблематичным;
* нет схематических обозначений, позволяющих отследить возможности возвратов между экранами;
* экраны уже изначально включают примеры UI, что при изменениях концепта, стиля или другой доработки придется редактировать всю карту – все ее элементы.

Во втором примере отсутствуют эти недостатки, и присутствуют дополнительные преимущества:

* есть возможность отследить все связи, понять насколько загружено каждое из окон;
* формат Иконка + Подпись – хороший способ для дизайнера понять на интуитивном уровне назначение окна.

**Монетизация**

Раздел содержит описание статей монетизации, как они работают, на кого нацелены, какой вклад в общий доход игры они внесут (в процентном соотношении), скидки, акции и прочее.

**Виральность**

Виральность – это характеристика приложения (в том числе, игрового), определяющая стремительность распространения контента среди пользователей. Если контент (текст, изображение, видео, аудиозапись и т.д) является виральным, он распространится в сети самостоятельно, силами пользователей.

В разделе описываются способы, которыми друзья приглашают своих друзей и то, почему эти способы работают; как это должно быть реализовано; какие есть оповещения и в каких случаях они появляются, их тексты; какие существуют бонусы, что и когда пишется на стены в соцсетях и т.д.