Оглавление

Благодарности

Я слышал, только авторы знают, каково это — написать книгу. Однако есть еще категория людей, которые знают точную цену, которую приходится заплатить, — те близкие люди, кому посчастливилось находиться рядом во время создания книги. Я писал книгу долго и могу признаться: это время было просто безжалостно украдено из нашей с моей женой Меган жизни. Мытье посуды и купание детей нельзя назвать «писательством», но без них данная книга не появилось бы.

Я начал работу над книгой, когда еще работал программистом в Electronic Arts. Вряд ли компания знала, как отнестись к моей затее, но я благодарен Майклу Мэлоуну, Оливеру Налле и Ричарду Вифаллу за их поддержку и подробные, проницательные отзывы о первых нескольких главах.

Когда книга была уже наполовину написана, я решил отказаться от услуг обычного издательства. Я знал, что тем самым у меня не будет профессионального редактора, но я получил десятки электронных писем от читателей, рассказывающих мне, какой они хотят видеть книгу. У меня также не было корректоров, но я получил более 250 замечаний и пожеланий по улучшению текста. И я с удовольствием отказался бы от работы по графику, но восторженные отзывы читателей после завершения очередной главы давали мне мотивацию работать дальше.

Некоторые называют подобные проекты «самиздат», но я предпочитаю «общественное издание». Может показаться, что написание книги — это работа одного

А вот что у меня было, так это хороший редактор. Лорен Бриз появилась именно тогда, когда я нуждался в ней, и она отлично справилась с работой.

Особая благодарность Колму Слоэну, который дважды тщательно перечитывал каждую главу в этой книге и оставлял невероятное количество фантастических советов и рекомендаций, и все от чистого сердца и совершенно бескорыстно. С меня пиво или двадцатка.

человека, но, трудясь над книгой, я не был одинок. Даже когда я отложил ее в долгий ящик, на целых два года, меня продолжали поддерживать. Без людей, неустанно напоминавших мне, что ждут новые главы, я никогда не взялся бы за книгу снова и не закончил бы ее.

Спасибо каждому, кто написал мне или оставил комментарий, оценил пост в интернете, рассказал о книге в твиттере или поделился постом, кто поддерживал со мной связь, или рассказал другу о книге, или отправил мне сведения об ошибке: мое сердце наполнено благодарностью вам. Создание книги было одной из главных целей в моей жизни, и вы помогли мне ее достичь. Спасибо!

Боб Нистрем, 6 сентября 2014 года

Введение

Часть I

Глава 1. Архитектура, производительность и игры

В пятом классе нам с друзьями дали ключи от небольшого школьного кабинета, временно не использовавшегося. Там стояло несколько очень старых компьютеров TRS-80. Надеясь вдохновить нас, учитель нашел распечатку нескольких простых программ на BASIC, с которыми мы могли бы повозиться.

Приводы для аудиокассет на компьютерах были сломаны, поэтому каждый раз, когда нам хотелось запустить какой-то код, нам приходилось тщательно его перепечатывать. Естественно, мы стали отдавать предпочтение программам всего в несколько строк:

```
10 PRINT "BOBBY IS RADICAL!!!"
20 GOTO 10
```

Возможно, если компьютер выведет этот текст достаточное количество раз, это магическим образом станет правдой.

Тем не менее процесс оказался непрост. Мы не знали, *как* программировать, поэтому крошечная ошибка в синтаксисе становилась фатальной для нас. Если программа не работала, а такое случалось частенько, нам приходилось приниматься за дело сначала.

В самом низу стопки, принесенной учителем, лежал настоящий монстр: программа, чей код занимал несколько страниц. Прошло немало времени, прежде чем нам хватило мужества хотя бы просто попробовать ее, но устоять мы не могли — над кодом значилось «Туннели и тролли». Мы понятия не имели, что это за программа, но так могла бы называться игра. А что может быть круче компьютерной игры, которую ты написал сам?

Мы так и не запустили ту программу, а через год нам пришлось освободить кабинет. (Много позже, когда я уже немного изучил BASIC, я понял: код представлял собой просто генератор персонажей для настольной игры, а не саму игру.) Но жребий был брошен — с тех пор я захотел стать разработчиком игр.

Когда я был подростком, в моей семье появился Macintosh с QuickBASIC, а позже THINK C. Я провел почти все свои летние каникулы, погрузившись в разработку игр. Самостоятельное обучение оказалось медленным и мучительным процессом. Я мог легко сделать что-нибудь работающее — например, экран с картой или простенькую головоломку. Но чем больше становилась игра, тем сложнее приходилось мне.

Правда, большую часть летних каникул я провел за ловлей змей и черепах в болотах южной Луизианы. И если бы не невыносимая жара, велик шанс того, что эта книга могла бы быть о герпетологии, а не о программировании.

Сначала я ставил перед собой задачу заставить работать хоть что-то. Затем мне захотелось писать программы по объему больше, чем могло уместиться у меня в голове. Вместо того, чтобы читать книги типа «Как программировать на C++», я начал искать книги об *организации* кода программы.

Быстрая перемотка на несколько лет вперед: друг вручает мне книгу *«Приемы объектно-ориентированного проектирования. Паттерны проектирования»*. Наконец-то! Книга, которую я искал с подросткового возраста. Я прочитал ее от корки до корки за один присест. Я все еще сталкивался с проблемами во время написания своих программ, но ведь всегда так приятно видеть, что другие люди тоже когда-то боролись с теми же проблемами и придумали решения. Я чувствовал: теперь в моих руках появились инструменты, и я могу больше не работать голыми руками.

Это была наша первая встреча, и через пять минут после того, как меня представили, я сел на кушетку и провел следующие несколько часов за чтением, полностью игнорируя нового знакомого. Хочется верить, что с тех пор мои социальные навыки немного улучшились.

В 2001 году я нашел работу своей мечты: инженер-программист в Electronic Arts. Я не мог дождаться момента, когда увижу настоящие игры и посмотрю на процесс их создания. Какая архитектура у такой огромной игры, как Madden Football? Как заставили взаимодействовать разные компоненты? Как смогли успешно выполнить один код на разных платформах?

Разбор кода стал для меня сложным, но удивительным опытом. Я видел потрясающий код графики, ИИ (искусственного интеллекта), анимации и визуальных эффектов. У нас работали люди, умевшие выжать каждый последний цикл из процессора и оптимизировать еще лучше. Вещи, о которых я даже не подозревал, оказались не просто *возможны*, эти люди успевали сделать их до обеденного перерыва.

Но *архитектура*, на которой основывался блестящий код, часто отодвигалась на второй план. Профи были так сосредоточены на новой функциональности, что организацию кода просто упускали из виду. Модули сильно друг от друга зависели. Новый функционал часто прикручивался к кодовой базе всюду, где только можно. К моему разочарованию, создавалось впечатление, будто эти программисты, подобно многим другим, никогда не слышали о *паттернах проектирования* или не продвинулись в чтении дальше паттерна Одиночка (Singleton).

Конечно, дела обстояли не так уж плохо. Раньше я представлял программистов в башне из слоновой кости, сидящих среди белых досок и, вооружившись фломастерами, неделями спокойно обсуждающих архитектуру. Реальность развеяла иллюзии: код, который я видел, был написан людьми, работающими в сжатые сроки. Они делали все возможное и, как я понял позже, выполняли свою работу на достаточно высоком уровне. Чем больше времени я тратил на код, тем больше я находил интересных решений, прятавшихся внутри.

К сожалению, именно «прятавшихся» — самое подходящее слово. В коде скрывались жемчужины, а люди просто проходили мимо, не замечая их. Я наблюдал, как коллеги изо всех сил пытались изобрести хорошие решения, когда то, что им было нужно, уже было написано до них в их же проекте.

Именно эта проблема и побудила меня к написанию данной книги. Я отобрал и отполировал лучшие образцы, которые я нашел в играх, и представил их здесь,

чтобы мы могли потратить время на изобретение чего-то нового, а не очередного *велосипеда*.

О чем эта книга

Итак, существует уже огромное количество книг по программированию игр. Зачем писать очередную? Да затем, что большинство книг по программированию, которые встречались мне, попадают в одну из двух категорий.

- **Узкоспециализированная литература**. Узко ориентированные книги. Они погружают вас в какой-то определенный аспект разработки игры. Учат вас 3D-графике, рендерингу в реальном времени, симуляции физики, искусственному интеллекту или работе со звуком. Это те области, на которых специализируются многие разработчики игр по мере развития их карьеры.
- **Книги о движках**. Эти, напротив, пытаются охватить все возможные аспекты всего игрового движка. Они ориентированы на создание полноценного движка, подходящего для определенного жанра игры, обычно 3D-шутера от первого лица.

Мне нравятся оба типа книг, но, я думаю, они кое-что упускают. Узкоспециализированные книги редко говорят вам, как один фрагмент кода взаимодействует с остальной частью игры. Вы можете быть мастером физики и рендеринга, но умеете ли вы изящно связать их?

Вторая категория рассказывает о связях элементов игры, но, на мой взгляд, книги, посвященные движкам, слишком однобокие и ориентированы на один-единственный жанр. А распространение мобильных и портативных игр ускорило развитие огромного количества различных жанров. Мы же не пытаемся сделать клон Quake. Книги, описывающие один движок, почти бесполезны, если *ваша* игра не подходит под его жанр.

Вот почему я попытаюсь работать больше в стиле *à la carte*. Каждая из глав моей книги является независимой идеей, которую вы можете воплотить в жизнь в своем коде. То есть вы можете выбирать и комбинировать их так, как вам нужно именно для *вашей* игры.

Другой пример стиля à la carte — популярная серия «*Жемчужины программирования игр*» (*Game Programming Gems*).

Как это относится к паттернам проектирования

Любая книга по программированию со словом «паттерн» в своем названии явно имеет отношение к классике «*Приемы объектно-ориентированного проектирования. Паттерны проектирования*» Эриха Гамма, Ричарда Хелма, Ральфа Джонсона и Джона Влиссидеса (также известных как «Банда четырех»).

Назвав книгу «Паттерны программирования игр», я не пытаюсь сказать, что книга «Банды четырех» неприменима к играм. Напротив, в разделе «Новый взгляд на паттерны проектирования» рассматриваются многие паттерны из «*Паттернов проектирования*», но акцент сделан именно на применении их в разработке игр.

Я даже, напротив, считаю, что книга применима не только к играм. Я мог бы также назвать ее «*Дополнительные паттерны проектирования*», но считаю игры наиболее подходящим и вдохновляющим примером. Или вы хотите прочитать еще одну книгу об учетных записях сотрудников и банковских аккаунтах?

И хотя приведенные здесь примеры полезны и при создании любого другого программного обеспечения, думаю, они особенно хорошо подходят для решения задач, обычно встречающихся при разработке игр.

В свою очередь, авторы «*Паттернов проектирования*» вдохновились другой книгой. Идея создания терминологии паттернов для описания найденных решений проблем появилась в «*Языке шаблонов*» Кристофера Александра (вместе с Сарой Исикава и Мюрреем Сильверштейном).

Их книга была посвящена архитектуре (*настоящей* архитектуре со зданиями и стенами и тому подобным), но они надеялись, что другие будут использовать ту же структуру для описания решений в других областях. «*Паттерны проектирования*» — попытка «Банды четырех» применить этот подход в программировании.

- Время и порядок следования часто являются основой архитектуры любой игры. Все должно происходить в правильном порядке и в нужное время.
- Сроки разработки сильно сжаты, и некоторым программистам необходима возможность быстро создавать и итерировать (изменять) огромное количество

различных поведений в игре, не наступая друг другу на пятки и не оставляя изменений по всему коду.

- После того как все поведения определены, начинается взаимодействие. Монстры бьют героя, зелья смешиваются, а бомбы взрывают и врагов, и друзей без разбора. Все взаимодействия должны реализовываться так, чтобы не превратить код в спутанный клок волос.
- И, наконец, производительность играет важную роль в играх. Разработчики игр соревнуются, желая выжать максимум из своей платформы. Владение приемами оптимизации определяет, чьи игры станут первоклассными и добьются миллионных продаж, а кому достанутся низкая частота кадров и сердитые пользователи.

Как читать эту книгу

«Паттерны программирования игр» разделены на три основные секции. Первая рассказывает о книге в общем. Это глава, которую вы сейчас читаете, а также следующая.

Второй раздел — он же вторая часть, *«Новый взгляд на паттерны проектирования»*, описывает несколько паттернов из книги «Банды четырех». В каждой главе я высказываю свое мнение о том или ином паттерне, а также о том, как именно он относится к разработке игр.

Последний раздел — настоящая суть книги. Он состоит из тринадцати паттернов проектирования, лично я нахожу их полезными при разработке игр. Они сгруппированы в четыре категории: «Паттерны очередности», «Паттерны поведения», «Паттерны уменьшения связанности» и «Паттерны оптимизации». При описании каждого из паттернов я старался придерживаться строгой структуры, чтобы вы могли использовать мою книгу в качестве справочника и быстро находить то, что вам нужно.

- Раздел **«Задача»** предоставляет краткое описание паттерна в рамках проблемы, которую можно

решить с его помощью. Он первый, что позволяет быстро найти в книге паттерн, подходящий для решения вашей проблемы.

- Раздел «**Мотивация**» описывает пример проблемы, к которой мы будем применять паттерн. В отличие от конкретных алгоритмов паттерн обычно кажется абстрактным, если не применяется к какой-либо конкретной проблеме. Обучение паттерну без примера — все равно что учить печь кекс, не используя теста. В этом разделе представлено тесто, а выпекают его в последующих разделах.
- Раздел «**Паттерн**» выделяет сущность паттерна из предыдущего примера. Если вы хотите получить сухое описание из учебника по паттернам, вот оно. Именно он вам нужен, если вы уже знакомы с паттернами и просто хотите немного обновить свои знания.
- До этого момента любой паттерн объясняется только на одном-единственном примере. Но как вы узнаете, подойдет ли паттерн для решения *вашей* задачи? Раздел «**Когда использовать**» содержит некоторые рекомендации о том, когда паттерн полезен и когда его лучше избегать. Раздел «**Имейте в виду**» указывает на последствия и риски при использовании паттерна.
- Если вам нужны конкретные примеры (мне всегда нужны), чтобы действительно понять теорию, тогда «**Пример кода**» — ваш раздел. Он шаг за шагом проходится через полную реализацию паттерна и дает полную картину того, как все работает.
- Паттерны отличаются от отдельных алгоритмов, потому что могут быть изменены. Каждый раз, когда вы используете паттерн, вы, скорее всего, реализуете его по-новому. В следующем разделе «**Архитектурные решения**» вы найдете различные варианты, на которые стоит обратить внимание при применении паттерна.
- В завершение есть короткий раздел «**Смотрите также**». Там показывается, как этот паттерн

соотносится с другими, а также дается ссылка на реальный проект с открытым кодом, в котором паттерн применяется.

О примерах кода

Примеры кода в книге написаны на языке C++, но это не означает, будто паттерны можно реализовать только на нем или что C++ для них лучше, чем другие языки. В принципе можно использовать абсолютно любой язык программирования, хотя некоторые паттерны подразумевают наличие в языке понятий объектов и классов.

Я выбрал C++ по нескольким причинам. Во-первых, это самый популярный язык для коммерческих игр. Это *lingua franca* отрасли. Более того, синтаксис C, на котором основан C++, лежит в основе Java, C#, JavaScript и многих других языков. Даже если вы не знаете C++, вы наверняка сможете понять приведенные примеры кода, приложив небольшие усилия.

Цель моей книги — *не* учить вас C++. Примеры максимально упрощены и не являются образцом хорошего использования C++. Рассматривая примеры, попытайтесь понять именно идею, которую выражает код, а не сам код.

В частности, код не написан в соответствии со спецификацией C++11 или более новой. Здесь не используются стандартные библиотеки, а шаблоны (templates) применяются редко. Все это приводит к «плохому» C++ коду в этой книге, но надеюсь, так он будет более доступным и понятным для людей, пришедших из C, Objective-C, Java и других языков.

Чтобы не тратить много места на код, который уже вам знаком или не имеет отношения к паттерну, его фрагменты иногда будут опущены. В коде место пропуска будет обозначено многоточием.

Рассмотрим функцию, которая выполняет некое действие, а затем возвращает значение. Пример касается только возвращаемого значения, а не выполняемой

работы. В нашем случае пример кода будет выглядеть так:

```
bool update()
{
  // Некие вычисления…
  return isDone();
}
```

Итак, что дальше

Паттерны — постоянно изменяющаяся и расширяющаяся область разработки программного обеспечения. В книге продолжается процесс, начатый «Бандой четырех» по документированию и совместному использованию паттернов проектирования, которые увидели они, и процесс будет продолжаться после того, как чернила высохнут на этих страницах.

Вы — основная часть этого процесса. Когда вы разрабатываете собственные паттерны и совершенствуете (или отклоняете!) предложенные, вы вносите вклад в сообщество разработчиков программного обеспечения. Если у вас есть предложения, замечания или любое другое мнение о написанном здесь, пожалуйста, свяжитесь со мной!