# Содержание (сводка)



# Содержание (настоящее)

## Введение

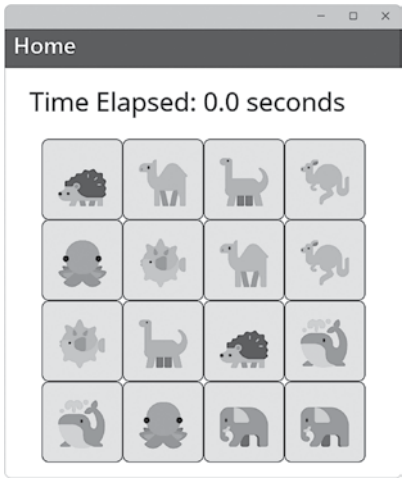
**Ваш мозг и C#.** Вы пытаетесь чему-то научиться, но ваш мозг «помогает» вам, делая все возможное, чтобы новые знания не закрепились. Он рассуждает примерно так: лучше оставить место для более важных вещей — например, поразмыслить о том, каких диких животных стоит избегать и плохая ли идея кататься на сноуборде голышом. Так как же обмануть мозг и заставить его думать, что от знания C# зависит ваша жизнь?

# 1 Начало работы с C#

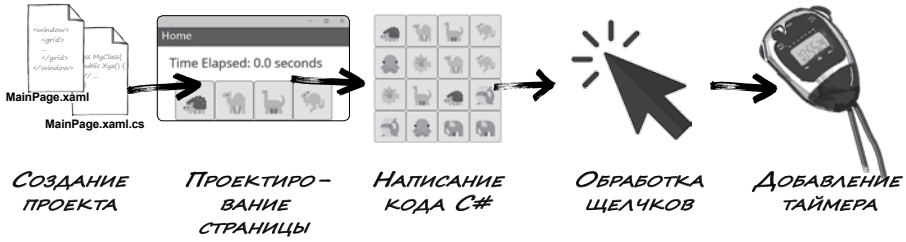
## Сделать что-то классное... и побыстрее!

**Хотите поскорее научиться программировать?**

C# — **современный и мощный** язык программирования. С **Visual Studio** вы получаете в распоряжение великолепную, интуитивно понятную среду разработки, которая старается сделать процесс программирования по возможности простым и приятным. Но Visual Studio — не только превосходный инструмент для написания кода, это также **чрезвычайно эффективный учебный инструмент** для освоения C#. Звучит заманчиво? **Тогда за дело!**





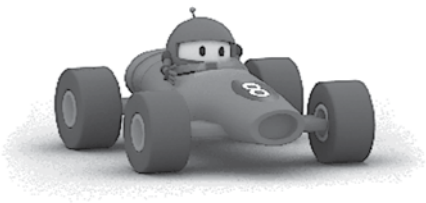
# 2 Переменные, команды и методы

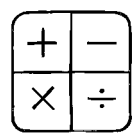
## Стремительный полет в C#

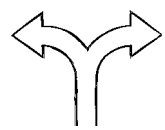
**Вы не просто пользователь IDE. Вы — разработчик.**

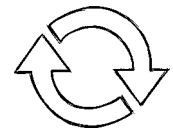
IDE может сделать за вас очень многое, и все же ее возможности не безграничны. Visual Studio — одна из самых совершенных систем разработки программного обеспечения, однако **мощная IDE** — только начало. Пришло время заняться **углубленным изучением кода C#**: какую структуру он имеет, как он работает, как управлять им... Потому что нет предела тому, что вы можете делать в ваших приложениях.





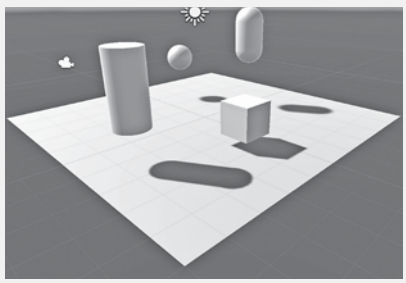


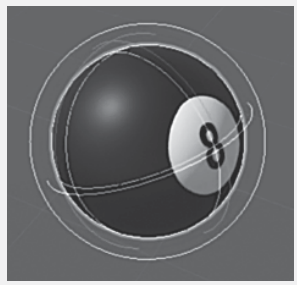
# Лабораторный курс Unity №1

## Исследование C# с Unity

Добро пожаловать на первый урок «**Head First C# Лабораторный курс Unity**». Написание кода — навык, и, как и любой другой навык, он развивается за счет **практики** и **экспериментирования**. И в этом отношении Unity может стать очень полезным инструментом. В этом лабораторном курсе вы сможете начать применять на практике то, что узнали о C# в главах 1 и 2.







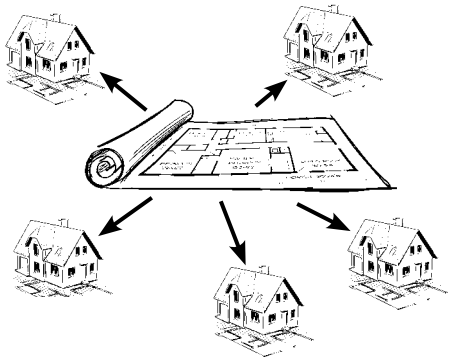
# 3 Пространства имен и классы

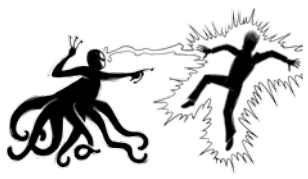
## Организация кода

**У хорошего разработчика код и данные всегда четко организованы.**

С чего начинается создание приложения? Вы прежде всего думаете, что **оно должно делать**, независимо от того, решаете вы научную задачу, создаете игру или просто развлекаетесь. Но не всегда очевидно, как отдельные команды вписываются в общую картину приложения... и тогда на помощь приходят **классы**. Они позволяют **структурировать код** вокруг создаваемой функциональности и задач, которые приложение должно решать. Классы также помогают **организовать данные** — они используются для создания **объектов**, представляющих любые сущности, используемые в вашей программе. Класс служит своего рода чертежом, по которому строятся объекты, используемые в приложении.





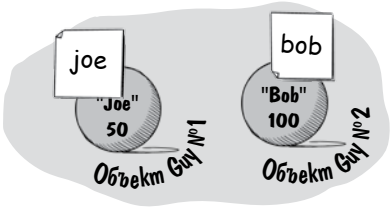


Данные, типы, объекты и ссылки

# 4 Управление данными приложения

**Данные и объекты — структурные элементы, из которых строятся ваши приложения.**

Чем были бы наши приложения без данных? Задумайтесь на минуту. Без данных наши программы... в общем, трудно представить, что кто-то станет писать код без данных. Вы запрашиваете **информацию** у ваших пользователей; эта информация используется для поиска данных или генерирования новой информации, которая возвращается пользователю. Собственно, практически все, что вы делаете в программировании, требует **работы с данными** в той или иной форме. В этой главе вы изучите все тонкости **типов данных и ссылок** C#, поймете, как работать с данными в программах, и даже узнаете кое-что новое об объектах (представьте, объекты — тоже данные!).

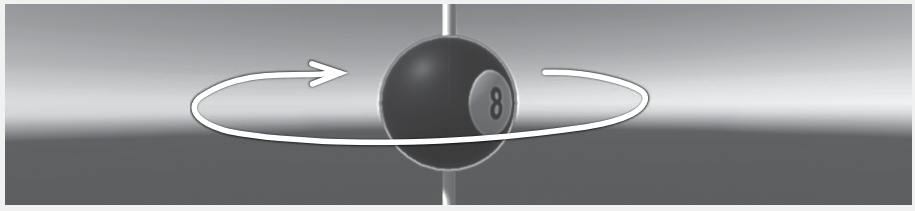




# Лабораторный курс Unity №2

## Написание кода C# для Unity

Unity — не только мощный кросс-платформенный движок и редактор для построения 2D- и 3D-игр и моделирования. Также это **отличный способ потренироваться в написании кода C#**. В этой лабораторной работе мы продолжим работу над проектом в Unity.

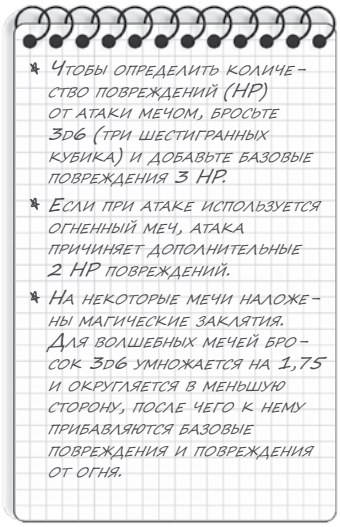


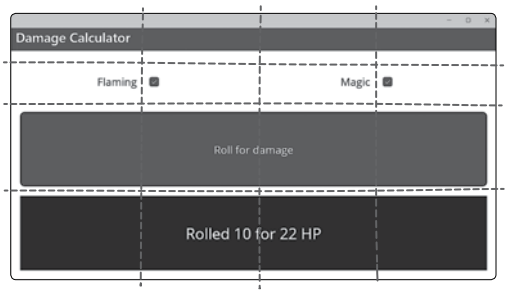


# 5 Инкапсуляция

## Как объекты хранят свои секреты

**Вам когда-нибудь хотелось, чтобы посторонние не лезли в ваши дела?** Вот и вашим объектам этого иногда хочется. И если вы не желаете, чтобы чужие люди читали ваш дневник или просматривали банковские выписки, хорошие объекты не позволяют другим объектам копаться в их полях. В этой главе вы узнаете о мощи инкапсуляции — механизме программирования, который делает ваш код более гибким. Такой код трудно использовать некорректно. **Данные вашего объекта объявляются приватными**, и к ним добавляются свойства, защищающие обращения к данным, — таким образом предотвращается **доступ к важным данным со стороны других объектов** и случайные злоупотребления ими.

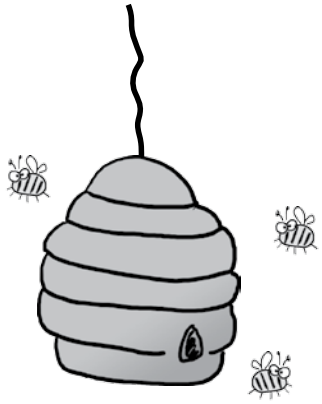



6 Наследование

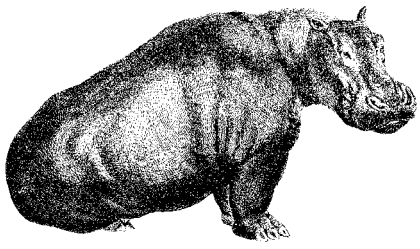
# Генеалогическое древо объектов

**Иногда люди ХОТЯТ быть похожими на своих родителей.**

Вы встречали класс, который действует почти так, как нужно? Думали ли вы о том, что при изменении ***всего*** нескольких элементов класс стал бы идеальным? **Наследование** позволяет расширить существующие классы, чтобы новый класс получал все поведение существующего, сохраняя при этом **гибкость** для внесения изменений, чтобы его можно было адаптировать под любые конкретные требования. Наследование является одним из самых мощных инструментов C#; в частности оно помогает **избегать дублирования кода**, более адекватно **моделировать реальный мир** и в конечном счете **упрощает сопровождение** и **снижает риск ошибок**.





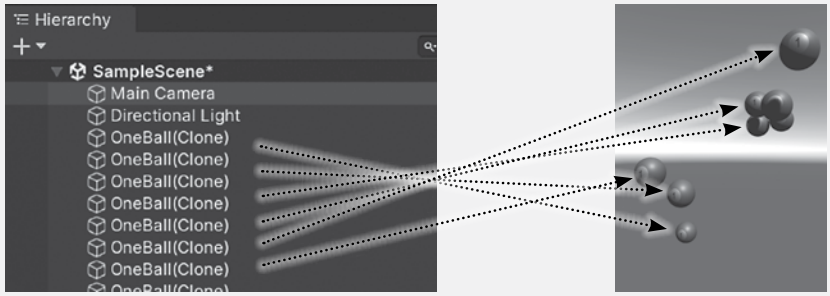


# Лабораторный курс Unity №3

## Экземпляры GameObject

C# — объектно-ориентированный язык, и поскольку эти лабораторные работы Head First C# Unity посвящены **практике написания кода на C#**, логично, что они будут сосредоточены на создании объектов.



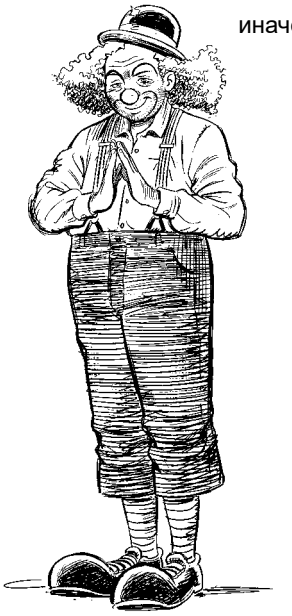


# 7 Интерфейсы, приведение типов и is

## Классы должны держать обещания

**Вам нужен объект для выполнения конкретной работы? Используйте интерфейс.**

Иногда возникает необходимость сгруппировать объекты по выполняемым ими функциям, а не по классам, от которых они наследуют. На помощь приходят интерфейсы. Интерфейсы могут использоваться для определения **конкретных задач**. Любой экземпляр класса, **реализующего** интерфейс, гарантированно выполняет эту задачу независимо от того, с какими другими классами он связан. Чтобы эта схема работала, каждый класс, реализующий интерфейс, должен гарантировать выполнение всех своих обязательств… иначе программа компилироваться не будет.





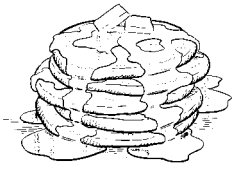
# 8 Перечисления и коллекции

## Организация данных

**Данные не всегда бывают такими аккуратными и ухоженными, как нам хотелось бы.**

В реальном мире данные, как правило, не хранятся маленькими аккуратными кусочками. Нет, данные поступают вагонами, штабелями и кучами. Для их систематизации нужны мощные инструменты, и тут вам на помощь приходят перечисления и коллекции. **Перечисления** — типы, позволяющие определять значения для классификации ваших данных. **Коллекции** — специальные объекты, способные **хранить и сортировать** данные, которые обрабатывает программа, и управлять ими. В результате вы можете сосредоточиться на основной идее программирования, оставив задачу управления данных коллекциям.

*Карта «Герцог быков». В природе не встречается.*





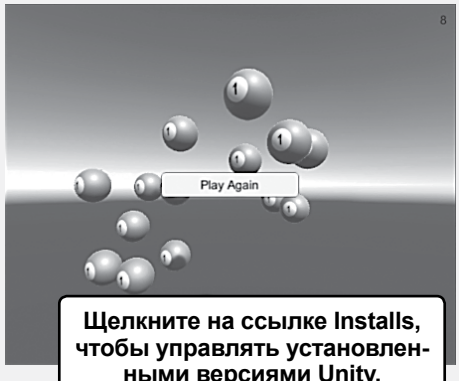
*По породе...*

# Лабораторный курс Unity №4

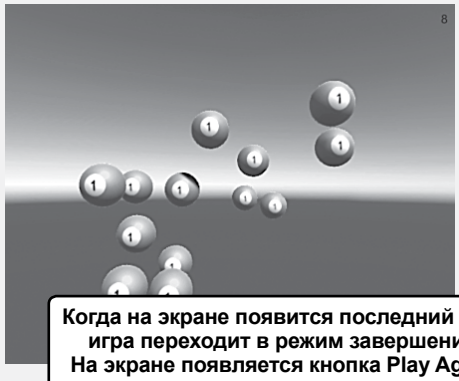
## Пользовательские интерфейсы

В предыдущей лабораторной работе Unity вы начали строить игру. Мы использовали заготовку для создания экземпляров GameObject, которые появлялись в случайных точках трехмерного пространства игры и летали по кругу. В этой лабораторной работе мы продолжим с того места, на котором остановились в предыдущей главе; в ней вы сможете применить то, что узнали об интерфейсах C#, и многое другое.






**Щелкните на ссылке Installs, чтобы управлять установленными версиями Unity.**



**Когда на экране появится последний шар, игра переходит в режим завершения. На экране появляется кнопка Play Again, и новые шары перестают появляться.**

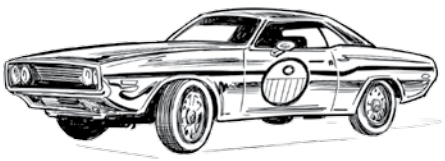
# 9 LINQ и лямбда-выражения

## Контроль над данными

**Этим миром правят данные… и нам нужно знать, как в нем жить.**

Прошли те времена, когда можно было программировать днями и даже неделями, не имея дела с **огромными объемами данных**. В наши дни данные стали **сутью любой программы**. LINQ — технология C# и .NET, которая позволяет не только **обращаться с запросами к данным** в коллекциях .NET на интуитивно понятном уровне, но и **группировать и выполнять слияние данных из разных источников. Модульные тесты** помогут убедиться в том, что ваш код работает так, как предполагалось. А когда вы освоитесь с задачей разбиения данных на блоки, с которыми удобно работать, вы сможете воспользоваться **лямбда-выражениями**, провести рефакторинг кода C# и сделать его еще более выразительным.



# 10 Чтение и запись файлов

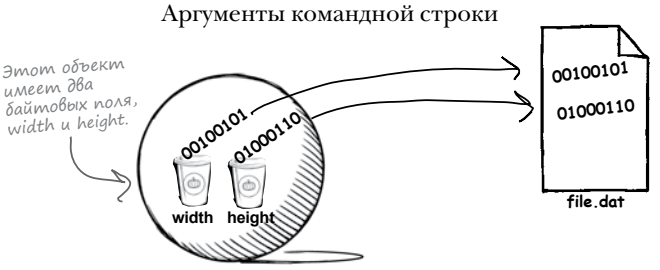
## Прибереги последний байт для меня

**Иногда долгосрочное планирование окупается.**

Пока что все ваши программы жили недолго. Они запускались, некоторое время работали и закрывались. Но этого недостаточно, когда имеешь дело с важной информацией. Вы должны уметь **сохранять свою работу**. В этой главе мы поговорим о том, как **записать данные в файл**, а затем о том, как **прочитать эту информацию**. Вы познакомитесь с **потоками данных**, узнаете о сохранении объектов в файлах с использованием **сериализации**, а также освоите работу с **шестнадцатеричными** и **двоичными** данными и кодировку **«Юникод»**.



```
0000: 45 6c 65 6d 65 6e 74 61 Elementa
0005: 72 79 2c 20 6d 79 20 64 ry, my d
0010: 65 61 72 20 57 61 74 73 ear Wats
0015: 6f 6e 21                on!
```

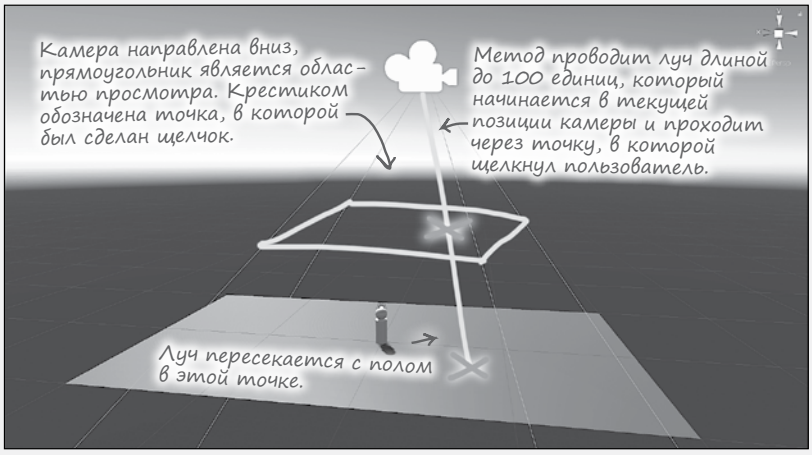


# Лабораторный курс Unity №5

## Отслеживание лучей

Создавая сцену в Unity, вы строите виртуальный 3D-мир, в котором перемещаются персонажи вашей игры. Но в большинстве игр объекты окружающей обстановки не контролируются игроком напрямую. Как же эти объекты определяют свое место в сцене? В этой лабораторной работе посмотрим, чем тут поможет C#.

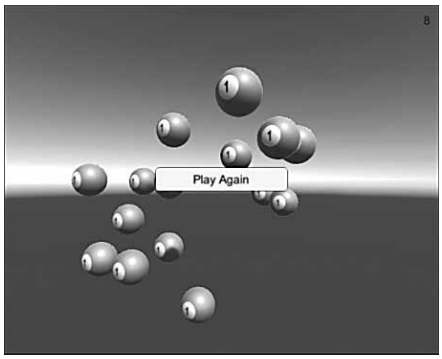




# CAPTAIN AMAZING

## СМЕРТЬ ОБЪЕКТА

| Head First C# | |
|---|---|
| Четыре доллара | Глава 11 |






# 12 Обработка исключений

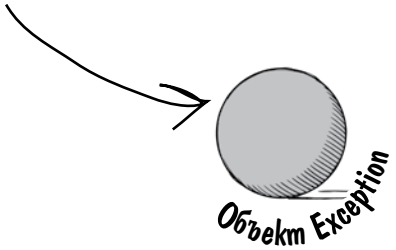
## Борьба с огнем надоедает

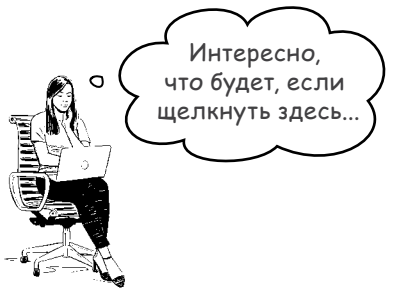
**Программисты не должны уподобляться пожарным.**

Представьте: прошло много лет. Вы усердно работали над развитием своих навыков C#, продолжали учиться и совершенствоваться и стали одним из ведущих разработчиков в крупной технологической компании. Но вам до сих пор продолжают звонить с работы по ночам, потому что **программа упала** или **работает не так, как должна работать.** Однако вы хотите проводить свое время за написанием кода, а не за тушением пожаров! Ничто так не выбивает из колеи, как необходимость устранять странные ошибки... но благодаря **обработке исключений C#** вы сможете написать код, который сам будет **разбираться с возможными проблемами**. А еще лучше, что вы можете планировать такие проблемы и **восстанавливать работоспособность программы** при их возникновении.




```
int[] anArray = {3, 4, 1, 11};
int aValue = anArray[15];
```



# Лабораторный курс Unity №6

## Навигация по сцене

В последней лабораторной работе Unity была создана сцена с полом (плоскость) и игроком (сфера с цилиндром). При этом использовался объект NavMesh, NavMesh Agent и отслеживание лучей, чтобы игрок следовал за щелчками в сцене. В этой работе мы дополним сцену с помощью C#.



**Это препятствие NavMesh создает движущийся проем в NavMesh, который мешает игроку перемещаться вверх по наклонной плоскости. Мы добавим сценарий, который позволяет игроку перетаскивать его мышью, чтобы блокировать и освобождать наклонную плоскость.**