Предисловие

Всегда мечтали создавать собственные игры? С книгой «Создаем игры и изучаем C++» ваша мечта может стать реальностью! Это удобное для начинающих руководство обновлено и улучшено с учетом новейших возможностей *Visual Studio 2022*, библиотеки *SFML (Simple Fast Multimedia Library)* и современных методов программирования на *C++20*. Вас ждет увлекательное погружение в мир программирования игр: вы создадите четыре полноценные игры — от простой до более сложной. Среди них — клоны таких популярных игр, как Timberman, Pong, шутер на выживание среди зомби и игра в жанре *бесконечный раннер*.

Книга начинается с основ программирования. Вы познакомитесь с такими ключевыми темами C++, как *объектно-ориентированное программирование (ООП)* и указатели C++, а также со *стандартной библиотекой шаблонов (STL)*. На примере игры Pong вы узнаете о методах обнаружения коллизий и основах игровой физики. В процессе разработки вы также изучите такие интересные концепции программирования игр, как массивы вершин, направленный звук, программируемые шейдеры OpenGL, генерация объектов и многое другое. Вы глубоко погрузитесь в игровые механики и реализуете обработку ввода, систему повышения уровня игрока и простой ИИ врагов. Наконец, вы познакомитесь с паттернами проектирования игр, которые помогут вам усовершенствовать навыки программирования на C++.

К концу книги вы получите необходимые знания для самостоятельного создания захватывающих игр с нуля.

Для кого эта книга

Книга идеально подойдет вам, если у вас нет никакого опыта в области программирования на C++, вам нужен курс для начинающих, чтобы освежить знания, вы хотите научиться создавать игры или просто изучить с их помощью C++.

Если вы стремитесь опубликовать игру (возможно, в Steam) или просто хотите поразить друзей, эта книга также окажется полезной.

Структура издания

Глава 1. Введение. Здесь описывается путь к написанию захватывающих игр для персональных компьютеров (ПК) с использованием C++ и SFML на базе OpenGL. Все основы C++, начиная с переменных, циклов, объектно-ориентированного программирования, STL, функций SFML и новых возможностей C++, были дополнены и расширены в этом издании. К концу книги вы не только создадите четыре полноценные игры, но и получите глубокие знания основ языка C++.

Глава 2. Переменные, операторы и условия: анимация спрайтов. В этой главе мы немного порисуем на экране: анимируем облака, которые будут двигаться со случайной скоростью и на разной высоте, а также пчелу, летящую на переднем плане. Для этого нам нужно будет изучить некоторые основы C++. Вы узнаете, как C++ хранит данные в переменных, как манипулировать этими переменными с помощью операторов и какие конструкции позволяют выполнять разные действия в зависимости от значения переменных. Затем вы сможете применить эти знания для реализации анимации облаков и пчелы.

Глава 3. Строки в C++, время в SFML, пользовательский ввод и HUD[1]. Примерно половину времени мы посвятим изучению работы с текстовыми данными и их отображению на экране, а оставшуюся часть — таймингу и созданию визуальной временной шкалы, которая будет побуждать игрока действовать быстрее.

Глава 4. Циклы, массивы, операторы switch, перечисления и функции: реализация игровых механик. В этой главе, пожалуй, содержится больше информации о C++, чем в любой другой. Она насыщена фундаментальными концепциями, которые значительно углубят ваше понимание языка. Здесь также объясняются некоторые ранее пропущенные сложные темы, такие как функции, игровой цикл и циклы в целом.

Глава 5. Коллизии, звук и условия завершения игры: приводим игру в состояние, чтобы в нее можно было полноценно играть. Это заключительный этап работы над первым проектом. К концу главы у вас будет первая полностью готовая игра. Как только вы запустите игру Timber!, обязательно прочитайте последний раздел данной главы, поскольку в нем будут предложены способы ее улучшения. Вот основные темы, которые мы охватим: добавление оставшихся спрайтов, обработка пользовательского ввода, анимация отлетающего бревна, обработка гибели персонажа, добавление звуковых эффектов и новых возможностей, а также улучшение игры.

Глава 6. Объектно-ориентированное программирование: приступаем к работе над игрой Pong. В этой главе будет немного теории, которая даст нам знания,

[1] HUD (англ. head-up display) — часть пользовательского интерфейса, где отображается важная информация о текущем игровом состоянии, например шкала здоровья, количество патронов, карта и т. д.

необходимые для ООП. ООП позволяет организовать код в понятные человеком структуры и лучше справляться со сложностью проекта. Мы не будем терять времени и сразу применим теорию на практике. Вы узнаете, как создавать новые типы данных в C++ и использовать их в качестве объектов, написав свой первый класс. Для начала мы рассмотрим упрощенный сценарий игры Pong, чтобы понять основы работы с классами, а затем создадим полноценную игру, применив изученные принципы.

Глава 7. AABB-метод обнаружения коллизий и физика: завершение работы над игрой Pong. Здесь мы напишем код нашего второго класса. Несмотря на то что мяч сильно отличается от ракетки, мы применим те же приемы, чтобы инкапсулировать внешний вид и функциональность мяча внутри класса `Ball`, как в случае с ракеткой и классом `Bat`. Затем мы напишем код для обнаружения коллизий и подсчета очков. Это может показаться сложным, но SFML значительно упростит задачу.

Глава 8. Использование области отображения и класса View в SFML: зомби-шутер. В этой главе мы еще больше погрузимся в ООП и познакомимся с классом `View`, который позволит разделить наш проект на слои для различных аспектов игры. Здесь мы реализуем слой для HUD и слой для основного игрового мира. Это необходимо, так как мир игры будет расширяться каждый раз, когда игрок уничтожает очередную волну зомби. В конце концов он станет больше экрана и игроку придется его прокручивать. Класс `View` поможет нам сделать так, чтобы текст HUD не двигался вместе с фоном.

Глава 9. Ссылки, спрайт-листы и массивы вершин в C++. В главе 4 мы говорили об области видимости. Это концепция, согласно которой переменные, объявленные внутри функции или блока кода, могут быть видны или использованы только в пределах этой функции или блока. Однако что будет, если нам нужно взаимодействовать с несколькими комплексными объектами в функции `main`? Получается, весь код должен быть в функции `main`? Но спасение есть.

В этой главе мы изучим ссылки в C++, которые позволят нам работать с переменными и объектами, находящимися вне области видимости. Кроме того, эти ссылки помогут избежать передачи больших объектов между функциями, что замедляет работу программы, поскольку копия переменной или объекта создается каждый раз.

Вооружившись новыми знаниями, мы рассмотрим в SFML класс `VertexArray`, который позволит быстро и эффективно отрисовывать на экране большие изображения, используя несколько частей одного графического файла. В результате с помощью ссылок и `VertexArray` к концу главы мы создадим масштабируемый, случайно генерируемый и прокручиваемый фон.

Глава 10. Указатели, стандартная библиотека шаблонов и управление текстурами. В этой главе мы узнаем много нового, а также продвинемся в разработке игры.

Мы изучим такую фундаментальную тему C++, как указатели. Это переменные, которые хранят адрес в памяти, обычно они содержат адрес другой переменной. Указатели немного напоминают ссылки, но они гораздо мощнее. Мы будем использовать их для работы с постоянно растущей ордой зомби.

Мы также познакомимся со стандартной библиотекой шаблонов (STL) — набором классов, позволяющих быстро и легко реализовать общие методы управления данными.

Глава 11. Класс TextureHolder и создание орды зомби. Разобравшись с основами STL, воспользуемся этими новыми знаниями для управления текстурами в игре. Если у нас будет 1000 зомби, вряд ли мы захотим, чтобы каждая копия зомби загружалась в память графического процессора (GPU).

Мы продолжим углубляться в ООП и применим статическую функцию — функцию, которую можно вызвать без создания экземпляра класса. Заодно узнаем, как спроектировать класс так, чтобы в нем существовал только один экземпляр. Это идеальный вариант, если разные части кода должны использовать одни и те же данные.

Глава 12. Обнаружение коллизий, бонусные предметы и пули. К этому моменту мы разработали основные визуальные аспекты игры: у нас есть игровой персонаж и арена, полная преследующих его зомби. Однако сейчас они не взаимодействуют друг с другом — зомби могут пройти сквозь игрока, не причинив ему вреда. Нам нужно реализовать обнаружение коллизий между зомби и игроком.

Если зомби получат способность наносить урон игроку, логично дать ему возможность защищаться. Поэтому мы вооружим его, чтобы он мог стрелять во врагов. Вдобавок ко всему, мы добавим класс для подбора аптечек и патронов.

Вот что мы реализуем в игре и в каком порядке: стрельбу пулями, прицел и скрытие указателя мыши, генерацию различных предметов, которые игрок сможет подбирать, и обнаружение коллизий.

Глава 13. Разделение области отображения на слои и реализация HUD. В этой главе мы увидим реальную ценность класса `View` библиотеки SFML. Мы добавим набор объектов класса `Text` и будем манипулировать ими, как делали это ранее в проектах Timber! и Pong. Новшеством станет создание HUD с помощью второго экземпляра `View`. Таким образом, HUD будет аккуратно располагаться поверх основной игровой сцены, независимо от того, что происходит с фоном, игроком, зомби и другими игровыми объектами.

Глава 14. Звуковые эффекты, работа с файлами и завершение игры. Мы почти закончили работу над проектом. В этой главе мы научимся управлять файлами, хранящимися на жестком диске, с помощью стандартной библиотеки C++, а также добавим звуковые эффекты. Конечно, мы уже знаем, как добавлять звуки, но здесь мы обсудим, где именно в коде будут находиться вызовы функции `play`.

Кроме того, мы устраним оставшиеся недочеты, чтобы игра выглядела завершенной. Мы рассмотрим следующие темы: сохранение и загрузка таблицы рекордов через ввод-вывод файлов, добавление звуковых эффектов, повышение уровня игрока и генерация новой волны врагов.

Глава 15. Игра Run. Добро пожаловать в финальный проект — игру Run. Это раннер, в котором цель игрока — бежать вперед по бесконечному уровню с исчезающими платформами и набрать как можно больше очков. В этом проекте мы изучим множество новых приемов программирования игр и еще больше тем по C++ для их реализации. Пожалуй, главное отличие этой игры от предыдущих заключается в том, что она будет гораздо более объектно-ориентированной. Мы создадим гораздо больше классов, но код для них будет лаконичным и несложным. Кроме того, мы построим игру так, что вся функциональность и внешний вид внутриигровых объектов будут сосредоточены в классах, а основной цикл игры останется неизменным, независимо от того, что делают игровые объекты. Это очень важно, поскольку в будущем позволит нам создавать разнообразные игры, просто проектируя новые компоненты (классы), описывающие поведение и внешний вид игровых объектов. Это значит, что вы сможете использовать одну и ту же структуру кода для создания совершенно другой игры по собственному замыслу. Но это еще не все — впереди нас ждет много интересного!

Глава 16. Звук, игровая логика, межобъектное взаимодействие и игровой персонаж. В этой главе вы узнаете, как легко и быстро реализовать звуковое сопровождение игры. Кроме того, всего за полдюжины строк кода мы добавим в игру музыку. Позже в проекте, но не в этой главе, добавим направленный (пространственный) звук.

После этого мы обернем весь наш код, связанный со звуком, в один класс под названием `SoundEngine`. Затем мы перейдем к созданию игрока. Мы реализуем всю функциональность основного персонажа, добавив два класса: один из них будет расширять класс `Update`, а другой — класс `Graphics`. Создание новых игровых объектов путем расширения этих двух классов станет основным подходом к добавлению новых элементов на протяжении всего проекта. Мы также разберем простой способ взаимодействия объектов друг с другом с помощью указателей.

Глава 17. Графика, камеры, действие. В этой главе мы подробно поговорим о графической составляющей проекта. Поскольку здесь мы будем писать код камер, выполняющих отрисовку, самое время рассмотреть и графику. Обратите внимание: в папке `graphics` всего один файл, и мы до сих пор не вызывали функцию `window.draw`. Мы обсудим, почему вызовы `draw` должны быть сведены к минимуму, а также реализуем классы `Camera`. Наконец, мы сможем запустить игру и увидеть камеры в действии: одну для общего вида, и одну для мини-карты.

Глава 18. Платформы, анимация игрового персонажа и элементы управления. В этой главе мы займемся программированием платформ, анимацией игрока и системой управления. На мой взгляд, самая сложная часть уже позади, и впереди нас ждет множество задач с более высоким соотношением вознаграждения к затраченным усилиям. Надеюсь, данная глава будет интересной: мы создадим

платформы, на которых будет стоять игрок, добавим ему функциональность, а также реализуем анимацию плавного бега персонажа через класс `Animator`.

Глава 19. Экран меню и дождь. Здесь мы реализуем две важные функции. Одна из них — экран меню, который будет информировать игрока о возможностях: начать, приостановить, перезапустить или завершить игру. Другая — создание эффекта дождя. Вы можете возразить, что дождь не нужен и не слишком подходит к игре, но это полезный навык, который стоит освоить. Самое интересное в главе — это то, как мы достигнем обеих целей с помощью классов, унаследованных от `Graphics` и `Update`. Мы объединим их в экземпляры `GameObject`, и они будут прекрасно работать вместе с другими игровыми сущностями.

Глава 20. Огненные шары и пространственный звук. В этой главе мы добавим все звуковые эффекты и HUD. Мы уже делали это в двух предыдущих проектах, но в этот раз все будет немного по-другому. Мы изучим концепцию пространственного звука и увидим, как SFML упрощает его использование. Кроме того, мы создадим класс для HUD, чтобы инкапсулировать код, отвечающий за вывод информации на экран.

Глава 21. Параллакс и шейдеры. К концу главы у нас будет полностью готовая и функциональная игра. Вот что мы сделаем, чтобы завершить работу над проектом: глубже погрузимся в OpenGL, шейдеры и язык GLSL (Graphics Library Shading Language), доработаем класс `CameraGraphics`, добавив шейдер и параллакс-эффект для фона, запрограммируем шейдер, взяв за основу сторонний код, и, наконец, запустим нашу игру.

Как извлечь максимум пользы из книги

Для работы с книгой не требуется никаких предварительных знаний. Вам не нужно уметь программировать — в процессе чтения вы сможете создать четыре полноценные игры. Если вы любите видеоигры и твердо намерены учиться — у вас все получится!

Загрузите файлы примеров кода

Примеры кода размещены на GitHub по адресу https://github.com/PacktPublishing/Beginning-C-Game-Programming-Third-Edition. У нас есть и другие наборы кода, доступные по адресу https://github.com/PacktPublishing/. Ознакомьтесь с ними!

Скачайте цветные изображения

Мы также предоставляем PDF-файл с цветными изображениями снимков экрана и диаграмм, используемых в книге. Вы можете загрузить его отсюда: https://packt.link/gbp/9781835081747.

Условные обозначения

В книге используется ряд типографских обозначений.

`Вот такой моноширинный шрифт` указывает на код в тексте, имена таблиц базы данных, папок и файлов, расширения файлов, наименования путей, фиктивные URL-адреса и пользовательский ввод. Например: «Мой основной каталог проекта — `D:\VS Projects\Timber`».

Блок кода задается следующим образом:

```
int playerScore = 0;
char playerInitial = 'J';
float valuePi = 3.141f;
bool isAlive = true;
```

Когда мы хотим обратить ваше внимание на определенную часть блока кода, соответствующие строки или элементы выделяются `шрифтом на сером фоне`:

```
// Сделайте спрайт дерева
Texture textureTree;
textureTree.loadFromFile("graphics/tree.png");
Sprite spriteTree;
spriteTree.setTexture(textureTree);
spriteTree.setPosition(810, 0);

while (window.isOpen())
{
```

Новые термины и важные слова выделены *курсивом*.

URL-адреса, слова, которые вы видите на экране, например в меню или диалоговых окнах, оформляются таким шрифтом и отображаются в тексте так: «Выберите пункт Информация о системе на панели Администрирование».

ПРИМЕЧАНИЕ

Предупреждения или важные заметки выглядят следующим образом.

СОВЕТ

Так обозначается совет или рекомендация.

Об авторе

Джон Хортон живет в Великобритании и увлекается программированием и играми. Он создает приложения, игры, пишет книги и ведет блог на тему программирования. Является основателем школы Game Code School.

Хочу посвятить эту книгу моим братьям, Рэю и Барри, за их наставления, пример и поддержку.

О научном редакторе

Йоан Рок — разработчик с более чем четырехлетним стажем в игровой индустрии. Обладая опытом программирования на C++, Йоан специализируется на разработке игр с использованием Unreal Engine и иногда Blueprints.

За время работы в Limbic Studio Йоан внес значительный вклад в разработку Park Beyond, AAA-игры, в которой игроки создают собственный парк развлечений и управляют им. Позже Йоан присоединился к Chillchat и работал над проектом Primorden — многопользовательской игрой на движке Unreal Engine 5 и системе Gameplay Ability. Он сыграл ключевую роль в реализации игровых механик, способностей монстров и деревьев поведения искусственного интеллекта (ИИ).

В Game Atelier Йоан возглавил разработку пользовательского интерфейса для еще не анонсированного проекта на основе Unreal Engine 5.3, Common UI и, конечно же, UMG.

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу comp@sprintbook.kz (издательство SprintBook, компьютерная редакция).

Мы будем рады узнать ваше мнение!