foxThread 7 окт 2022 в 02:03

C++ и Raylib как глоток свежего воздуха

10 мин 13K

C++*

Из песочницы

Предыстория

Какое то время назад пришла ко мне идея исполнить хотя бы немного детскую мечту программировать игры. Надо сказать, что определенный опыт у меня был. Попала в девятом классе ко мне в руки чудесная книга Андрэ Ла мота "Секреты программирования игр", благодаря которой я изучал язык Си, поскольку все там было завязано на нем. И это были незабываемые моменты. Просто это казалось каким то чудом, что я сам, своими руками заставляю двигаться персонажей и вообще, это все даже похоже на игры для Dendy, которые я так мечтал делать в еще более ранний период детства.

В основном сейчас я программирую на Java, и изначально мой выбор пал на библиотеку libGDX. Масштабные фреймворки типа Unity и UnrealEngine я не рассматривал,поскольку главной целью моей все же было не делать продукт, а просто получить удовольствие от написания игры и ощутить ту магию, когда в школе на языке Си я с нуля писал код для доступа в видеопамять, рисования спрайтов и контролировал каждый бит (почти). Поэтому просто хотелось поэкспериментировать именно с библиотеками, предоставляющими основные функции, типа рисования, обработки клавиатуры. Но как то попалась мне на глаза библиотека Raylib. Испугал меня конечно язык C++.(К слову сказать Raylib поддерживает еще и кучу других языков, но это я узнал потом). Долго ли коротко, принял решение поэкспериментировать с Raylib, заодно изучив C++,на котором никогда не программировал и вообще для меня C++ это какой то высший пилотаж.

Эпизод 1. C++

Для того, чтобы начать, необходимо конечно же разобраться с C++. Последние впечатления от языка Си (именно Си) у меня были только в школе, но начав разбираться, я понял, что старая закалка дает о себе знать и все не так уж и сложно. Да и вообще перейти с Java на Си++ по части синтаксиса не так и сложно. Да и принципы объектно-ориентированного программирования они и в Африке принципы. Однако все оказалось не совсем так как я представлял себе. И главное коварство состояло в том, что думать, что C++ это просто язык с немного отличающимся синтаксисом от Java -очень большое заблуждение. Столкнулся я с несколькими ситуациями, которые ну никак не мог понять.

Например, такая история:

```
#include "spaceship.h"

void main(){

Spaceship ship;
cout << "Ok";

}
```

При компиляции и запуске программа внезапно завершалась и не доходила до вывода на консоль. Будете смеяться, но я потратил пол дня на поиск проблемы, изломал всю голову, пытаясь ответить на вопрос : "Где? и почему?". Почему, когда я убирал объявление переменной ship,все работало? Для программиста C++ мои страдания вызовут лишь недоуменную ухмылку, но я то думал, что я все еще на планете Java. И это просто объявление указателя на объект типа Spaceship,под который даже память то еще не выделена. Но в C++ это означает, что создается объект, выделяется память и вызывается конструктор без параметров! Такие дела. Эта привычка считать все указателями изрядно потрепала мне нервы в начале экспериментов с C++.

Ну или например ключевое слово this. Ну почему например this.x не работает? Да потому что this это указатель на объект и по синтаксису C++ правильно писать this->x . Поэтому в итоге я понял одно: Программируя на C++,программируй на C++, а не думай, что ты пишешь на Java с измененным синтаксисом.

В общем, решил я применить свой способ изучения нового языка, фреймворка по излюбленной схеме. Поняв только самую базу, идти в бой в реальный проект, пытаясь хоть и неумело, изобретая кучу велосипедов но писать так, как подсказывает логика, интуиция и здравый смысл. Суть в том, что в процессе такой работы я обычно прихожу к действительно правильному пониманию концепций и способов использования языка. Сама жизнь говорит(возвращаясь к примеру):" Не помещай инициализацию объектов прямо в конструктор, лучше вызови потом функцию init(), когда будешь готов :-)"

Я заведомо не читал различные мануалы о том, как правильно организовать игровой цикл, как делать анимацию, как грузить спрайты, потому что мне было просто интересно придумать это, пусть на примитивном уровне, но самому. Может быть это ностальгия по детству, когда единственное что я прочитал в книжке это как вывести картинку в память видеокарты и обработать клавиатуру. Конечно, основные концепции были мне ясны, вроде этапов игрового цикла, но более детальные алгоритмы рождались моей фантазией. И библиотека Raylib как раз очень мне подходит, поскольку там именно основные возможности, типа нарисовать текстуру, проверить клавиатуру, а не реализованный движок с искусственным интеллектом , анимацией. Ну что же...вперед.

Эпизод 2. Установка необходимых инструментов

Сначала конечно необходим компилятор. К слову говоря при скачивании Raylib с официального сайта предлагается несколько сборок с включенными в них компиляторами. Я остановился на сборке с классическим компилятором GCC и его портом на Windows MinGw-w64. После установки главное прописать путь PATH к каталогу BIN компилятора и можно приступать к сборке .

В школе я долгое время пользовался средой Borland C++, где компиляция осуществлялась в самой среде и я плохо осознавал, на самом деле, что же происходит когда я вижу надпись: Build Succesful. Позже, столкнувшись с нехваткой памяти в реальном режиме MSDOS, а ее катастрофически не хватало, нашел на одном из дисков дистрибутив компилятора Watcom C++ 10.0. Данная штука среды не имела, зато позволяла работать со всей имеющейся памятью свыше 640 килобайт при помощи расширителя(так вроде его называли) DOS4GW.

Пришлось мне тогда изучать концепции того, как происходит вообще сборка exe, как прилинковать библиотеку и т.д. И еще тогда мне было страшно смотреть на попадающиеся файлы с названием Makefile, потому что лишь чутка заглянув в их содержимое мне становилось плохо от супер непонятной структуры их текста. Надо сказать, тот опыт очень сильно мне помог сейчас, чтобы как то сдвинуться с мертвой точки и знать от чего вообще плясать. На самом деле, все не так уж и страшно. Итак, вкратце, имеем следующую структуру проекта:

```
mygame
    Include
    Obj
    Res
    Src
    Makefile
```

В инструментах GCC утилита make. Если брать аналог из мира Java,это что - то отдаленно напоминающее maven или gradle. И даже принцип такой же, мы просто запускаем из каталога проекта команду MAKE и она ищет в текущем каталоге Maefile и выполняет прописанные в нем инструкции по сборке исполняемого файла. Для понимания давайте разберем некоторые строчки этого чудесного файла.

```
RAYLIB_PATH= e:/tools/raylib/raylib
```

Здесь мы задаем константу, где указываем путь к библиотеке RayLib. Надо учесть, что помимо самой библиотеки в скачанном дистрибутиве содержится и компилятор MinGW и различные инструменты, поэтому у нас и идут два каталога RayLib подряд.

В файле в любом месте задаются константы, которые мы можем использовать в пределах его объема. Например:

```
OBJ_DIR=OBJ
```

```
SRC_DIR=SRC
```

Здесь мы задаем каталоги, где хранятся исходные файлы с расширением .CPP, а также каталог OBJ, куда будут сохраняться скомпилированные исходные файлы. Любой исходный файл компилируется в машинный код и сохраняется в так называемый объектный файл. Обычно они имеют расширение .o. Правило, превращающее каждый .CPP в в .O имеет следующий вид:

```
$(OBJ_DIR)/%.o:$(SRC_DIR)/%.cpp Makefile
        g++ -c -o $@  $^ $(CFLAGS)  -MMD $(INCLUDE_PATHS) -D $(PLATFORM)
```

Не пугайтесь, это выглядит страшно, а на самом деле все очень просто и даже тривиально. Наиболее важные моменты таковы:

```
$(OBJ_DIR)/%.o:$(SRC_DIR)/%.cpp
```

здесь мы просто говорим, что каждый объектный файл формируется из исходного файла с тем же названием. Обратите внимание на использование заданных выше констант. То есть если в каталоге SRC два файла main.cpp и game.cpp,то в каталоге OBJ в результате работы этого правила появиться два файла main.o и game.o.

```
g++ -c -o $@  $^ $(CFLAGS)  -MMD $(INCLUDE_PATHS) -D $(PLATFORM)
```

это собственно команда компиляции, мы используем программу g++(это компилятор C++) . Интересные моменты в двух последовательностях $@ и $^. Это просто подстановки, где первая заменяется на левую часть предыдущего выражения, а вторая на правую. Очень удобно для задания в командной строке компилятора конечного файла и исходного. Можно представить себе, что выполняется цикл по всем CPP и мы каждую итерацию получаем новый исходник и компилируем его. Стоит также обратить внимание на константы CFLAGS и INCLUDE_PATHS.

CFLAGS - это просто строка, где мы собрали всякие разные параметры компилятора. Вот как она выглядит у нас:

```
CFLAGS = -Wall -std=c99 -D_DEFAULT_SOURCE -Wno-missing-braces -Wunused-result
```

INCLUDE_PATHS -набор каталогов, откуда компилятор подключает заголовочные файлы .h. Дело в том, что встречая в коде программы инструкцию #include, компилятор просто включает его содержимое в текущий файл. Это просто подстановка текста. Можно включить в Ваш код любой файл с кодом на C++,и использовать его классы и функции. Однако на практике в заголовочных

файлах не содержится сам код, а лишь определения функций и классов. Когда компилятор встречает в исходном тексте создание объекта определенного класса, он смотрит, есть ли в области видимости определение класса, и если находит его, то считает, что все в порядке. Сам код реализации класса содержится например в файле CPP и компилируется ОТДЕЛЬНО. На последней стадии сборки исполняемого файла, программа, называемая линковщик собирает весь скомпилированный код вместе.

Сама библиотека Raylib упрощенно говоря представляет собой два файла:

- libraylib.a - это раз таки скомпилированный код с библиотекой
- raylib.h - заголовочный файл.

  В нашем makefile есть строчка, подключающая отдельные библиотеки из кода libraylib.a:

```
LDLIBS = -lraylib -lopengl32 -lgdi32 -lwinmm
```

Все! Абсолютно ничего лишнего и сложного. И это в какой то степени заставляет меня восхищаться C++.

Немного хотелось бы отвлечься от технических деталей и рассказать о своей проблеме в программировании, с которой я столкнулся еще в школе. Все шло хорошо, пока я экспериментировал в программировании под DOS. Все что я писал ложилось в стройную и понятную систему. Я понимал механизм работы программы до самого последнего бита и пиксела. Наверное каждая строчка отражалась в сознании коротким и ясным отрывком, ударом барабана, и сотни этих строчек составляли в неокрепшем детском мозгу цельную картину мира. Наверное в этом я реализовывал свою потребность в контроле. Потом время пришло время, когда я познакомился с C++ Builder и вообще стал пытаться разобраться с программированием под Windows. И тут моя стройная картина мира стала рушиться. Я просто не понимал, как это все работает. То есть вся фишка в том, что разобраться в том, как программировать GUI под Windows не составляло проблемы, просто сам факт, что теперь мне непонятно ЧТО ЭТО за штуковина такая, действия которой я должен отслеживать функциями обратного вызова, теперь мне непонятно КАК этот проект на C++ Builder запускается, где у него точка входа и что происходит, когда я двойным щелчком по кнопке открываю функцию OnCLick(). Уровень абстракции увеличивался, и от этого страх неизвестности также прогрессировал. То есть те инструменты, которые по идее должны упрощать жизнь программиста и увеличивать его эффективность, меня отталкивали. И на какое то время я вообще забросил свои изыскания в программировании. Именно из за этой потери контроля. Конечно, сейчас я смотрю на это с улыбкой, но в свое время меня это очень напрягало. Ну и если честно посмотреть, на самом деле уровень абстракции функции printf отличается от набора нулей и единиц точно так же, как уровень flutter отличается от SkiA.

На самом деле, сегодня мне важно понимать, что сложность современного программирования заставляет доверять все большим и большим аксиомам, выраженных в различных фреймворках, фреймворках над фреймворками и т.д. И нет нужды понимать всю матрешку абстракций, это просто непозволительно в современных условиях. Но все же... вот эти эксперименты с RayLib, - тот самый глоток свежего морозного воздуха.

Эпизод 3 Игровой цикл

Перейдем к собственно коду, друзья.

Хочу порассуждать, что же такое Игра? Игра это система сущностей, действующих по определенным алгоритмам. Тут как нельзя кстати фраза-вся наш жизнь -Игра!. Или весь наш мир-игра. И недаром наверное часто употребляется в игровой индустрии словосочетание игровой мир. Игра представляет собой набор циклов, в которых совершаются повторяющиеся действия. И вообще, любую игру от танчиков до последней какой-нибудь супер навороченной виртуальной системы можно описать следующим образом:

- В каждый момент времени игровая система пребывает в определенном состоянии, которое складывается из состояний ее частей. Сюда входят, к примеру:
  - расположения танков, зданий, людей
  - координаты пуль, летящих из снарядов, или гриба, убегающего от Марио:-)
  - состояние убежищ, урон главного героя, текущая высота полета ниндзя в прыжке
  - и много много много различных параметров.
- Получаем информацию от контроллеров ввода
- Определенный алгоритм рассчитывает состояние всех частей системы в данный момент
- Происходит отрисовка той части мира, которая необходима игроку (по мнению создателей игры). В Марио это кусочек уровня, в танчиках на Dendy вовсе статичный экран, а в шутере от третьего лица вид из глаз главного героя
- Все повторяем опять и опять.

  Вот вкратце как это реализуется в небольшом эксперименте, который я написал при помощи средств библиотеки Raylib.

```
while (!WindowShouldClose())

//Начинаем бесконечный цикл до возникновения события ЗАКРЫТЬ_ОКНО

if(IsKeyPressed(KEY_LEFT) || IsKeyPressed(KEY_RIGHT)){

// Получаем данные от контроллеров ввода

      reaper.startWalkForth(); //изменяем состояние
}

 BeginDrawing();  //собственно, производим отрисовку

    ClearBackground(RAYWHITE);
    reaper.draw();

 EndDrawing();
```

Да. это весь код. Лукавлю конечно. Дьявол в деталях, но принципиально ничего сложного нет. В реализованном мной примере нет логики именно игрового мира, то есть никто не стреляет, не надо обсчитывать не попала ли пуля в врага и не нанесла ли ему урон, сколько энергии осталось у героя, нашел ли он ключ к двери и т.д. Здесь больше логики, направленной на решение двух простых задач:

- В каком месте экрана изобразить главного героя
- Какой кадр из его анимационных последовательностей нарисовать

Класс AnimationCycle

Для этого я написал класс AnimationCycle,являющийся моделью одной анимационной последовательности. Например, герой идет вперед. Данному действию соответствует объект walkForth. Соответственно для реализации удара и прыжка создаются объекты kicking,jumpStart и jumpLoop. Для загрузки спрайтов я написал метод, который загружает из определенной папки все лежащие там картинки формата PNG, исходя из того, что каждая картинка это отдельный кадр.

Сами спрайты я скачал на ресурсе где лежат кучи бесплатных и не бесплатных картинок. Таких ресурсов куча по поиску free sprites,например. Спрайты хранятся в объектах типа Texture и загружаются вполне себе очевидным и однозначным способом.

```
Image image=LoadImage(path.c_str()); //Загружаем картинку из файла
ImageResize(&image,width,height); // можно изменить ее размеры(отмасштабировать)

Texture frame=LoadTextureFromImage(image); //Загрузка текстуры
```

Хотелось бы обратить внимание на один момент, который попил у меня крови. Если мы начнем загружать текстуры до инициализации системы openGL, то программа вывалится. И это логично. Потому что мы не просто загружаем текстуру, мы загружаем ее в память видеокарты, реализующей интерфейс openGL .Вот эти две строчки инициализируют систему openGL.

```
    InitWindow(screenWidth, screenHeight, "MyGame");
    SetTargetFPS(60);
```

Класс Reaper

Данный объект хранит в себе все возможные состояния анимации героя и соответствующую логику управления ими.
Основные моменты, на мой взгляд, требующие внимания:

- Игровой цикл производит отрисовку всего экрана 60 раз в секунду, однако мы можем захотеть не менять кадры так быстро. К примеру, наш герой идет вперед и мы хотим, чтобы он шел со скоростью 24 кадра в секунду. Если мы будем менять кадр каждую итерацию игрового цикла, то ничего у нас не получиться, поэтому мы заводим внутри объекта специальный таймер, который обнуляем после завершения прорисовки очередного кадра и запускаем опять.
- Я разбил выполняемую в данный момент анимацию на две составляющих: Анимация движения и Анимация действия:

```
class Reaper{
   AnimationCycle *currentWalkingAnimation;
   AnimationCycle *currentActionAnimation;
}
```

Для чего это мне? Например, герой идет вперед и одновременно бьет палкой. При этом его currentWalkingAnimation=walkForth, а currentActionAnimation=kicking. На время того, как наше создание бьет палкой, его анимация ходьбы прерывается и начинает исполняться анимация удара. По достижении последнего кадра анимации удара, опять идет отрисовка ходьбы. Для разделения этих двух видов анимаций, я завел свойство infiniteAnimation. Если для определенной анимации infiniteAnimation равен TRUE, то остановить ее можно лишь принудительно, например когда игрок отжал клавишу вправо.
Если же FALSE,то анимация просто прекращается . Данная логика реализована в методе draw() класса Reaper.

В данной статье я не претендую на правильность или неправильность выбранных методик и способов реализации игрового цикла. Просто очень хотелось поделиться тем опытом, который есть у меня со всеми странностями и где то откровенными штуками, наверное вызывающими улыбку у более опытных профи. Весь код есть на GitHub.

.Надеюсь, кому то эта статья окажется полезной.

Теги: изучение нового, c++, game development, интересные истории

Хабы: C++

@foxThread

Пользователь

Подписаться

Комментарии 12

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

neohmd 9 часов назад

Как проектируют дата-центры? Разбираемся на практике

10 мин 3.3K

Обзор

+24 28 2 +2

Wonderlove вчера в 21:19

Как создать свой стикер пак с помощью нейросети Stable Diffusion

Простой 9 мин 9.1K

Туториал

+20 88 11 +11

popski_ruvds 5 часов назад

Памяти Кевина Митника — хакера, ломавшего ФБР, АНБ и Кремниевую долину. Часть 9: туман Сиэтла, саспенс и чёрный вертолёт

Простой 10 мин 1.2K

Ретроспектива

+19 8 0

 **Bright_Translate** 9 часов назад

Как уничтожить вашу ОС с помощью TAR

Средний 12 мин 5.2K

Кейс Перевод

+17 25 10 +10

 **muxa_ru** 19 часов назад

Так ли плох отечественный софт, как его описывают в комментах?

2 мин 7.6K

+14 15 49 +49

 **GeeksCat** 7 часов назад

Dangerous Dave: история похождений Дейва или тот, кто сам стучится в дверь

Простой 9 мин 1.5K

Ретроспектива

+12 8 2 +2

 **Travis_Macrif** 10 часов назад

Как синий экран смерти остановил работу аэропортов и предприятий

7 мин 3.3K

Ретроспектива

+11 8 15 +15

 **divolko3** 10 часов назад

Объясни свою маленькость: пять ну очень миниатюрных ноутбуков

4 мин 6.6K

+9 9 12 +12

 **crackidocky** 7 часов назад

Быстрый поиск по разной документации

Простой 2 мин 1.1K

Туториал

+7 21 6 +6

patnashev 7 часов назад

Большие простые числа: теория и практика их поиска

8 мин 1.4K

Из песочницы

+6 10 5 +5

Геймифицируй, защищай: как в ЕВРАЗе запустили приложение для охоты на риски

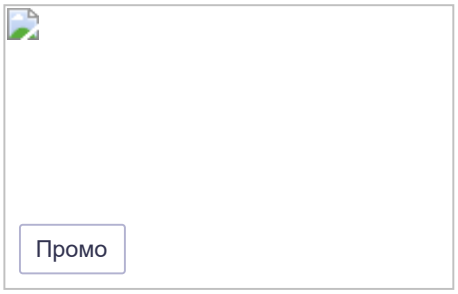
Турбо

Показать еще

МИНУТОЧКУ ВНИМАНИЯ

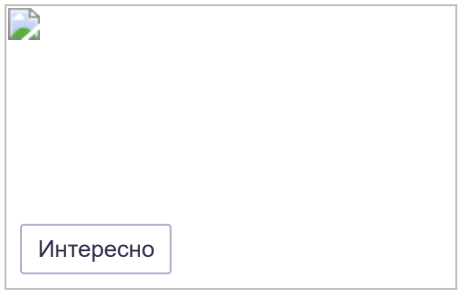
Турбо

3D-модели по промту и роботы: что было на IT-ивенте GigaConf



Промо

Есть подозрение, что вам недоплачивают?



Интересно

Соединяем тех, кому нужен совет, с теми, у кого есть опыт

ЗАКАЗЫ

VUE мини-PWA для новичка

1000 руб./за проект · 5 откликов · 47 просмотров

Плагин для iiko front (.Net 4.7.2) для коммуникации с 1с

75000 руб./за проект · 2 отклика · 22 просмотра

Разработка 3 персонажей и Нфт коллекции

1111 руб./за проект · 2 отклика · 38 просмотров

Аудит Телеграм канала

1000 руб./за проект · 4 отклика · 29 просмотров

(доработать) HTML переходник в стиле BrawlStars

1000 руб./за проект · 6 откликов · 65 просмотров

Больше заказов на Хабр Фрилансе

ЧИТАЮТ СЕЙЧАС

Марсоход НАСА «Кьюриосити» обнаружил сюрприз в марсианском камне

36K 24 +24

Две ночи полной Луны: 20/21 и 21/22 июля 2024 года

37K 20 +20

Дискеты начинают и выигрывают: флот Германии до сих работает с экзотическими 8-дюймовыми флоппи-дисками

41K 85 +85

Как уничтожить вашу ОС с помощью TAR

5.2K 10 +10

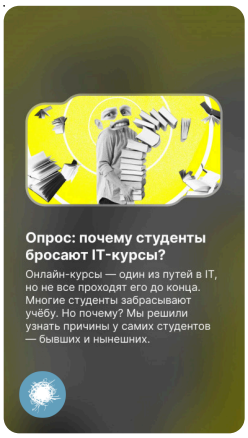
Браузер Chrome будет просить пользователя указать причину загрузки подозрительного файла

7.1K 72 +72

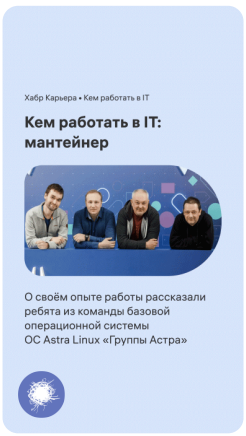
Геймифицируй, защищай: как в ЕВРАЗе запустили приложение для охоты на риски

Турбо

ИСТОРИИ



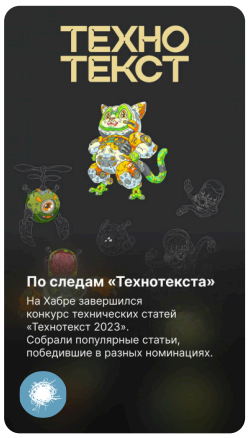
Почему люди бросают курсы



Кем работать в IT: мантейнер

Активность найма во 2 квартале 2024

Зарплаты в IT во первой половине 2024



Победители Технотекста 2023

Годно компа

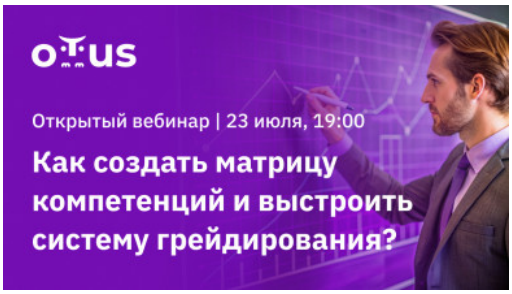
РАБОТА

QT разработчик
6 вакансий

Программист C++
123 вакансии

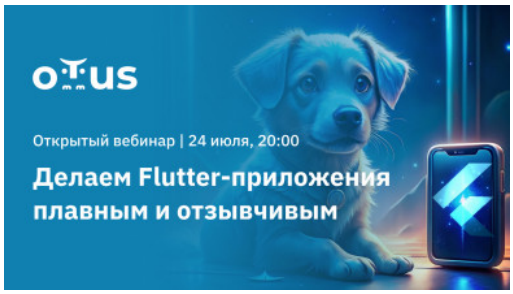
Все вакансии

БЛИЖАЙШИЕ СОБЫТИЯ



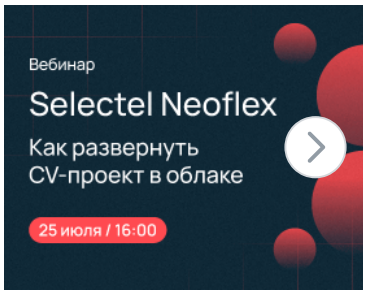
23 июля

Вебинар «Как создать матрицу компетенций и



24 июля

Вебинар «Делаем Flutter-приложения плавными и



25 июля

Вебинар «Как пост MLOps-конвейер д

выстроить систему грейдирования»

Онлайн

Менеджмент Тестирование

Больше событий в календаре

отзывчивыми»

Онлайн

Разработка

Больше событий в календаре

проекта в облаке»

Онлайн

Разработка

Администрирование А

Больше событий в календар