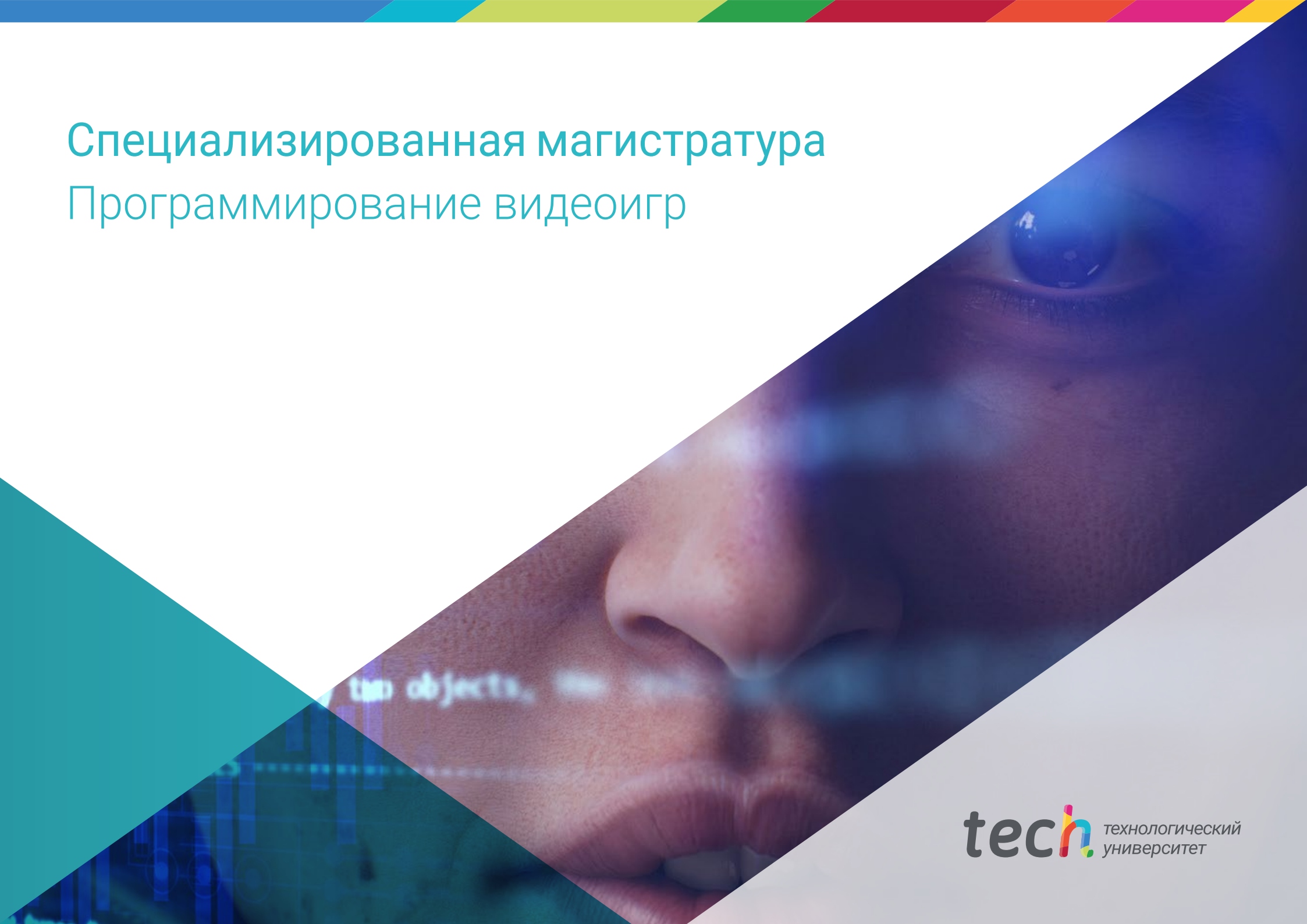
Специализированная магистратура

Программирование видеоигр



tech технологический университет

tech технологический университет

Специализированная магистратура

Программирование видеоигр

- Формат: **онлайн**
- Продолжительность: **12 месяцев**
- Учебное заведение: **TECH Технологический университет**
- Режим обучения: **16ч./неделя**
- Расписание: **по своему усмотрению**
- Экзамены: **онлайн**

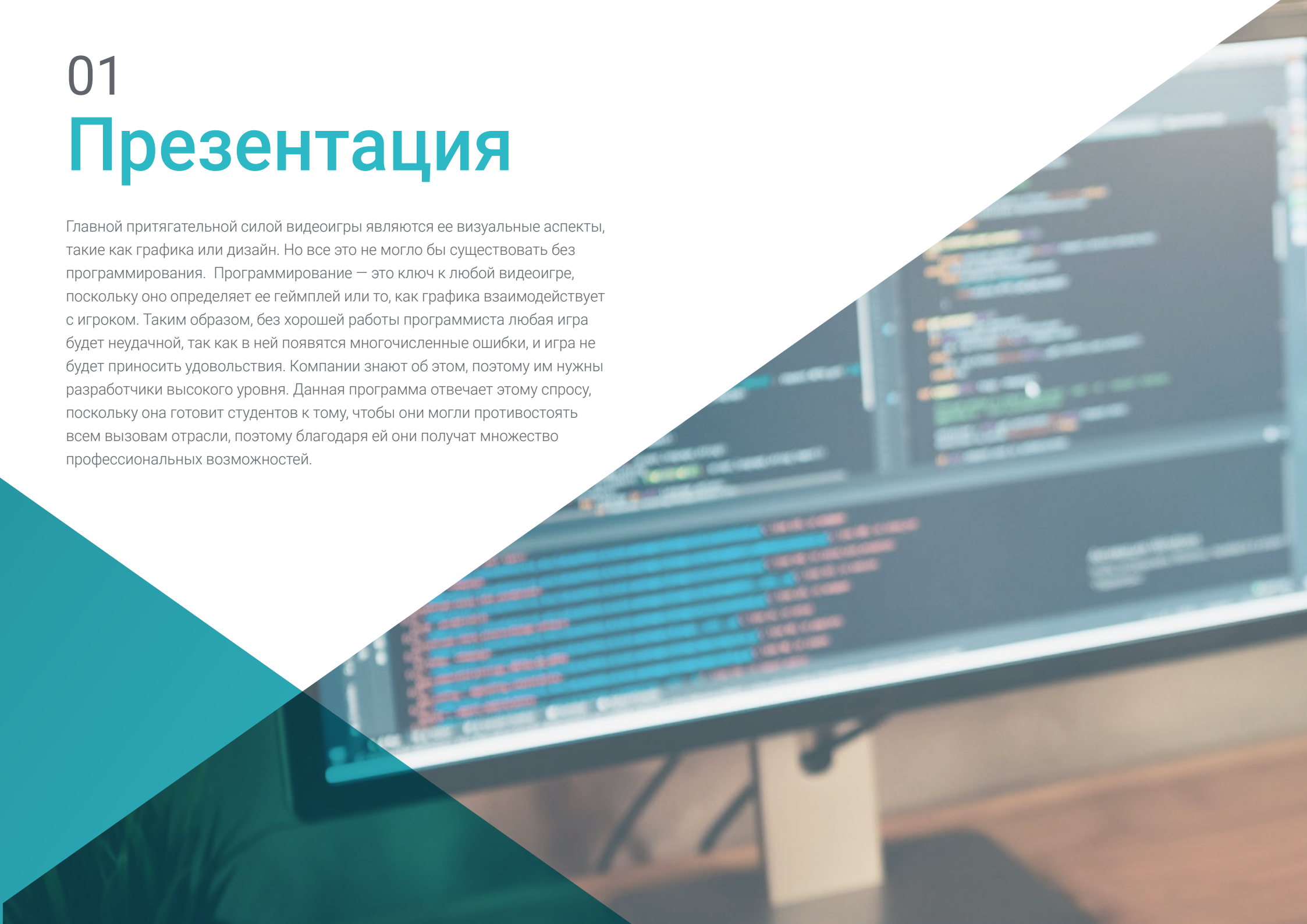
Веб-доступ: www.techtitute.com/ru/information-technology/professional-master-degree/master-video-game-programming

Оглавление

01

Презентация

Главной притягательной силой видеоигры являются ее визуальные аспекты, такие как графика или дизайн. Но все это не могло бы существовать без программирования. Программирование — это ключ к любой видеоигре, поскольку оно определяет ее геймплей или то, как графика взаимодействует с игроком. Таким образом, без хорошей работы программиста любая игра будет неудачной, так как в ней появятся многочисленные ошибки, и игра не будет приносить удовольствия. Компании знают об этом, поэтому им нужны разработчики высокого уровня. Данная программа отвечает этому спросу, поскольку она готовит студентов к тому, чтобы они могли противостоять всем вызовам отрасли, поэтому благодаря ей они получат множество профессиональных возможностей.



Компании знают, что главное в видеоигре — это программирование. Специализируйтесь в этой области и станьте самым востребованным разработчиком в своей среде"

За каждой великой видеоигрой стоит огромная команда профессионалов, специализирующихся в каждой области работы, которые постараются сделать свою компанию успешной. Обычно наиболее заметными для фанатов являются те разделы, которые они могут воспринимать непосредственно, например, визуальная часть или разделы, связанные с управлением персонажем, механикой или взаимодействием с объектами.

Однако, чтобы все эти элементы работали и были правильно интегрированы, есть одна важная задача, которую часто не учитывают, — программирование. Разработка видеоигры имеет несколько этапов и включает в себя различные отделы, но именно программирование придает всему этому смысл и формирует основной скелет, на который нанизываются все остальные области.

Именно поэтому компании в этой отрасли уделяют этому вопросу так много внимания, так как знают, что правильная и эффективная разработка их видеоигр облегчит продвижение проекта и позволит избежать появления ошибок и *багов*. Именно поэтому они ищут лучших программистов, специализирующихся в этой области.

Но найти настоящих специалистов в этой области не так уж и просто. И данная Специализированная магистратура в области программирования видеоигр отвечает этому спросу, позволяя студентам стать отличными экспертами в разработке видеоигр, которые могут с легкостью развиваться в индустрии, получая большие возможности для карьерного роста благодаря навыкам и умениям, которые они приобретут на протяжении этой программы.

Данная **Специализированная магистратура в области Программирование видеоигр** содержит самую полную и современную образовательную программу на рынке. Основными особенностями обучения являются:

- Разбор практических кейсов, представленных специалистами в области программирования и разработки видеоигр
- Наглядное, схематичное и исключительно практичное содержание курса предоставляет научную и практическую информацию по тем дисциплинам, которые необходимы для осуществления профессиональной деятельности
- Практические упражнения для самооценки, контроля и улучшения успеваемости
- Особое внимание уделяется инновационным методологиям
- Теоретические занятия, вопросы эксперту, дискуссионные форумы по спорным темам и самостоятельная работа
- Учебные материалы курса доступны с любого стационарного или мобильного устройства с выходом в интернет

Разрабатывайте все виды видеоигр в лучших компаниях мира благодаря данной Специализированной магистратуре"

Программирование становится все более важной сферой в разработке видеоигр. С этой программой вы станете более востребованным специалистом в отрасли"

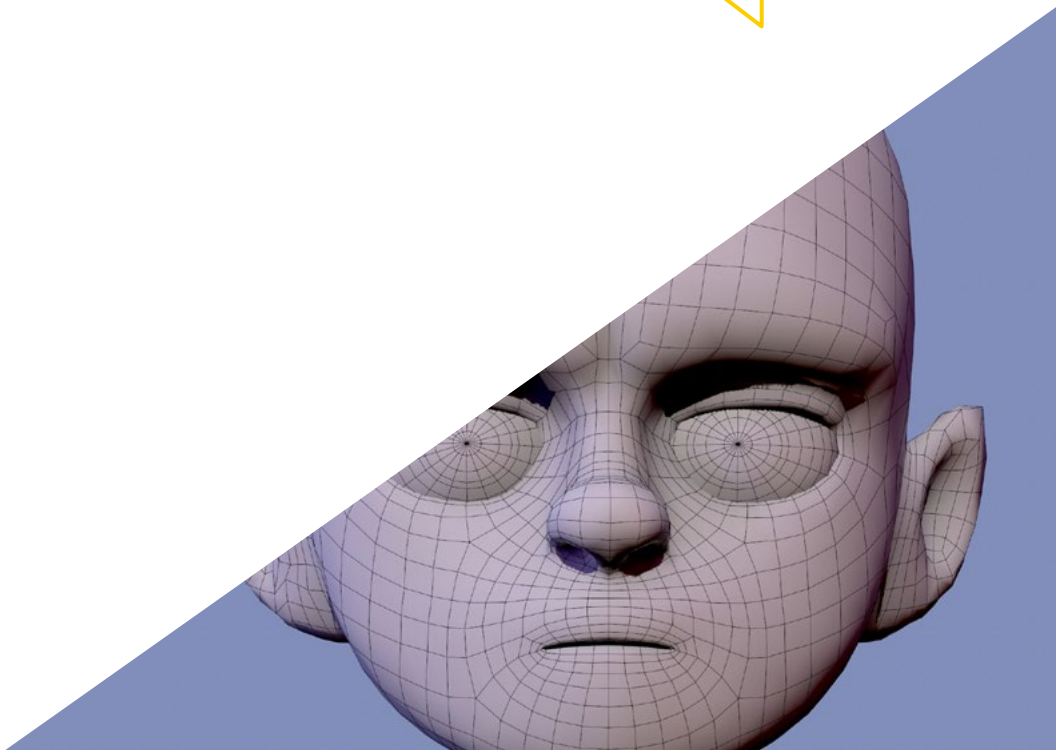
Игры — ваша страсть, и вы хотите стать великим разработчиком. Не раздумывайте и записывайтесь в данную Специализированную магистратуру.

Лучшие компании в отрасли ждут вас. Специализируйтесь сейчас.

В преподавательский состав программы входят профессионалы отрасли, признанные специалисты из ведущих сообществ и престижных университетов, которые привносят в обучение опыт своей работы.

Мультимедийное содержание программы, разработанное с использованием новейших образовательных технологий, позволит специалисту проходить обучение с учетом контекста и ситуации, т.е. в симулированной среде, обеспечивающей иммерсивный учебный процесс, запрограммированный на обучение в реальных ситуациях.

Структура этой программы основана на проблемно-ориентированном обучении, с помощью которого специалист должен попытаться решить различные ситуации из профессиональной практики, возникающие в течение учебного курса. В этом поможет инновационная интерактивная видеосистема, созданная признанными экспертами.



02

Цели

Основная цель данной Специализированной магистратуры — сделать из студентов отличных разработчиков видеоигр. Эта индустрия расширяется и нуждается во все большем количестве программистов и специалистов с высоким уровнем подготовки, поэтому эта программа идеально подходит для получения отличных карьерных возможностей в одних из самых престижных компаний в мире. Таким образом, данная программа предлагает своим студентам все необходимые навыки, чтобы стать востребованными специалистами в этом секторе, добиваясь значительного и немедленного продвижения по карьерной лестнице.

Все ваши мечты теперь в пределах вашей досягаемости благодаря данной Специализированной магистратуре в области программирования видеоигр"

Общие цели

- Знать различные языки программирования и методы, применяемые в видеоиграх
- Углубиться в процесс создания видеоигр и в интеграцию программирования на этих этапах
- Изучить основы разработки видеоигр и теоретические знания, которые должен знать разработчик видеоигр
- Освоить основные языки программирования, используемые в видеоиграх
- Применить знания в области инженерии ПО и специализированного программирования к видеоиграм
- Понять роли программирования в разработке видеоигр
- Знать о различных существующих консолях и платформах
- Разработать веб-игры и многопользовательские видеоигры

Когда вы пройдете обучение по этой программе, вы станете лучшим разработчиком видеоигр в своем окружении"

Конкретные цели

Модуль 1. Основы программирования

- Понимать базовую структуру компьютера, программное обеспечение и языки программирования общего назначения
- Анализировать основные элементы компьютерных программ, такие как различные типы данных, операторы, выражения, операторы ввода-вывода и управляющие операторы
- Интерпретировать алгоритмы, которые являются необходимой основой для разработки программного обеспечения

Модуль 2. Структура данных и алгоритмы

- Изучить основные стратегии проектирования алгоритмов, а также различные методы и меры для вычисления
- Различать функционирование алгоритмов, их стратегию и примеры их использования в основных известных задачах
- Освоить технику *Backtracking* и ее основные способы применения

Модуль 3. Объектно-ориентированное программирование

- Студенты познакомятся с различными паттернами проектирования для объектно-ориентированных задач
- Понять важность документации и тестирования при разработке программного обеспечения
- Управлять использованием потоков и синхронизации, а также решать общие задачи в параллельном программировании

Модуль 4. Консоли и игровые устройства

- Знать базовые операции основных периферийных устройств ввода и вывода
- Понимать основные преимущества дизайна различных платформ
- Изучить структуру, организацию, функционирование и взаимосвязи устройств и систем
- Понять роль операционной системы и пакетов разработки для мобильных устройств и игровых платформ

Модуль 5. Программная инженерия

- Различать основы программной инженерии, а также процесса и различные модели разработки программного обеспечения, включая agile-технологии
- Ознакомиться с проектированием требований, их разработкой, составлением, согласованием и проверкой, чтобы понимать основные стандарты, связанные с качеством программного обеспечения и управлением проектами

Модуль 6. Движки для разработки игр

- Узнать о функционировании и архитектуре движка видеоигры
- Понимать основные особенности существующих игровых движков
- Программировать правильно и эффективно приложения, применяемые к игровым движкам
- Выбрать наиболее подходящую парадигму и языки программирования для программирования приложений, применяемых к игровым движкам

Модуль 7. Интеллектуальные системы

- Изучить понятия, связанные с теорией агентов, архитектурой агентов и процессом их рассуждений
- Усвоить теоретическую и практическую информацию, лежащую в основе концепции информации и знаний, а также различные способы представления знаний
- Понимать, как функционируют семантические анализаторы, системы, основанные на знаниях, и экспертные системы

Модуль 8. Программирование в реальном времени

- Проанализировать ключевые особенности языка программирования реального времени, которые отличают его от традиционных языков программирования
- Понимать основные концепции информационных систем
- Научиться применять основные подходы и методы программирования в реальном времени

Модуль 9. Проектирование и разработка веб-игр

- Студент сможет разрабатывать игры и интерактивные веб-приложения с соответствующей документацией
- Оценивать основные особенности игр и интерактивных веб-приложений, чтобы коммуницировать в своей области профессионально и корректно

Модуль 10. Многопользовательские сети и системы

- Описать архитектуру протокола управления передачей/интернет-протокола (TCP/IP) и основные принципы работы беспроводных сетей
- Проанализировать способы обеспечения безопасности, применяемые в видеоиграх
- Научиться разрабатывать многопользовательские онлайн-игры

03

Компетенции

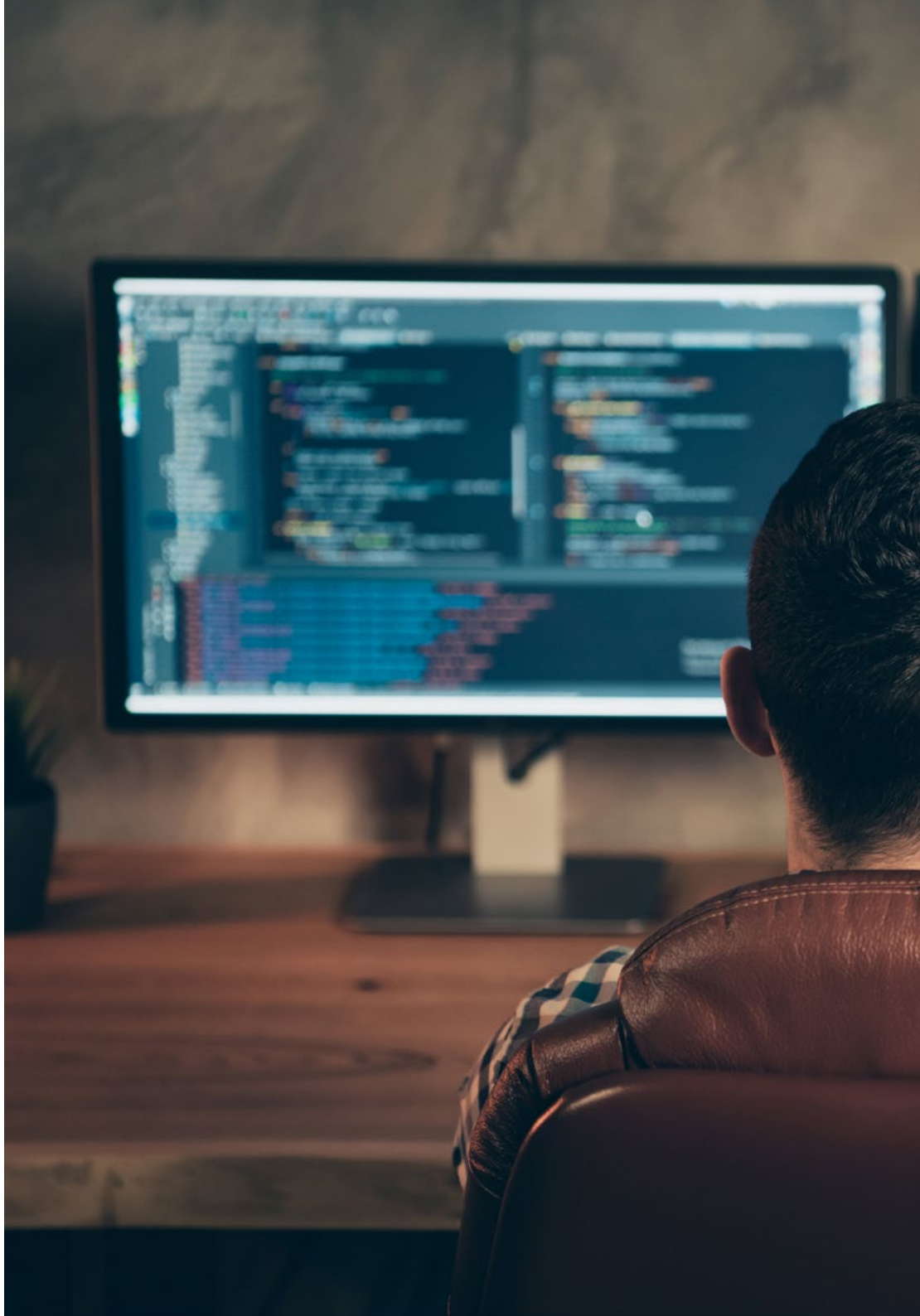
Данная Специализированная магистратура в области программирования видеоигр превращает студентов в настоящих специалистов по разработке этого типа аудиовизуальных проектов благодаря компетенциям и навыкам, которые она им дает. Таким образом, благодаря этой замечательной программе студенты получат ряд профессиональных инструментов, с помощью которых они смогут решать любые задачи, связанные с программированием видеоигр, став, таким образом, незаменимыми специалистами для своих компаний.

Вы освоите все аспекты разработки видеоигр"

Общие профессиональные навыки

- Разработать все этапы видеоигры, от первоначальной идеи до финального запуска
- Получить специализацию в качестве программиста видеоигр
- Внимательно изучить все этапы разработки: начиная с базовой архитектуры, программирования персонажа игрока и всех элементов, задействованных в игровом процессе
- Получить общее видение проекта, уметь предложить решения различных проблем и задач, возникающих при разработке видеоигры

Достигните совершенства в качестве программиста видеоигр благодаря данной Специализированной магистратуре"



Профессиональные навыки

- Знать программное обеспечение, необходимое для того, чтобы стать профессиональным разработчиком видеоигр
- Понимать опыт игрока и уметь анализировать игровой процесс
- Понимать все теоретические и практические части процесса программирования видеоигр
- Освоить самые полезные языки программирования для вселенной видеоигр
- Интегрировать изученные способы программирования в различные виды консолей и платформ
- Программировать веб-игры и многопользовательские видеоигры
- Усвоить концепцию игрового движка, чтобы уметь правильно программировать
- Применять знания в области разработки программного обеспечения в разработке видеоигр

04

Структура и содержание

Содержание данной Специализированной магистратуры в области программирования видеоигр было тщательно разработано командой выдающихся специалистов в этой области, которые в совершенстве знают современное состояние отрасли. Таким образом, благодаря этой программе студенты смогут получить все необходимые знания, чтобы иметь возможность отвечать на запросы компаний данного сектора, поскольку они были специально подготовлены с учетом их особенностей и специфики, которые являются сложными и постоянно меняющимися.

Эти материалы сделают вас отличным экспертом в программировании видеоигр"

Модуль 1. Основы программирования

1.1. Введение в программирование
- 1.1.1. Базовая структура компьютера
- 1.1.2. Программное обеспечение
- 1.1.3. Языки программирования
- 1.1.4. Жизненный цикл программного приложения

1.2. Разработка алгоритмов
- 1.2.1. Решение задач
- 1.2.2. Описательные техники
- 1.2.3. Элементы и структура алгоритма

1.3. Элементы программ
- 1.3.1. Происхождение и характеристики языка C++
- 1.3.2. Среда разработки
- 1.3.3. Концепция программы
- 1.3.4. Фундаментальные типы данных
- 1.3.5. Операции
- 1.3.6. Выражения
- 1.3.7. Операторы
- 1.3.8. Ввод и вывод данных

1.4. Операторы управления
- 1.4.1. Операторы
- 1.4.2. Бифуркации
- 1.4.3. Петли

1.5. Абстракция и модульность: Функции
- 1.5.1. Модульное программирование
- 1.5.2. Понятие функции и полезности
- 1.5.3. Определение функции
- 1.5.4. Поток выполнения во время вызова функции
- 1.5.5. Прототип функции
- 1.5.6. Возвращение результатов
- 1.5.7. Вызов функции: Параметры
- 1.5.8. Передача параметров по ссылке и по значению
- 1.5.9. Область идентификатора

1.6. Статические структуры данных
- 1.6.1. Arrays
- 1.6.2. Матрицы. Полиэдры
- 1.6.3. Поиск и сортировка
- 1.6.4. Строки. Функции ввода/вывода для строк
- 1.6.5. Структуры. Объединения
- 1.6.6. Новые типы данных

1.7. Структуры динамических данных: Указатели
- 1.7.1. Понятие. Понятие указателя
- 1.7.2. Операторы и операции с указателями
- 1.7.3. Массивы указателей
- 1.7.4. Указатели и массивы
- 1.7.5. Указатели на строки
- 1.7.6. Указатели на структуры
- 1.7.7. Множественная косвенность
- 1.7.8. Указатели на функции
- 1.7.9. Передача функций, структур и массивов в качестве параметров функции

1.8. Файлы
- 1.8.1. Основные понятия
- 1.8.2. Операции с файлами
- 1.8.3. Типы файлов
- 1.8.4. Организация файлов
- 1.8.5. Введение в файлы C++
- 1.8.6. Работа с файлами

1.9. Рекурсия
- 1.9.1. Определение рекурсии
- 1.9.2. Виды рекурсии
- 1.9.3. Преимущества и недостатки
- 1.9.4. Соображения
- 1.9.5. Рекурсивно-итеративное преобразование
- 1.9.6. Стек рекурсии

1.10. Тестирование и документация
- 1.10.1. Тестирование программ
- 1.10.2. Тестирование методом "белого ящика"
- 1.10.3. Тестирование методом "черного ящика"
- 1.10.4. Инструменты для тестирования
- 1.10.5. Программная документация

Модуль 2. Структура данных и алгоритмы

2.1. Введение в шаблоны разработки алгоритмов
- 2.1.1. Рекурсия
- 2.1.2. "Разделяй и властвуй"
- 2.1.3. Другие стратегии

2.2. Эффективность и анализ работы алгоритмов
- 2.2.1. Меры эффективности
- 2.2.2. Измерение объема данных на входе
- 2.2.3. Измерение времени выполнения
- 2.2.4. Случаи: худший, лучший и средний
- 2.2.5. Асимптотическая нотация
- 2.2.6. Критерии математического анализа нерекурсивных алгоритмов
- 2.2.7. Критерии математического анализа рекурсивных алгоритмов
- 2.2.8. Эмпирический анализ алгоритмов

2.3. Алгоритмы сортировки
- 2.3.1. Концепция сортировки
- 2.3.2. Пузырьковая сортировка
- 2.3.3. Сортировка выбором
- 2.3.4. Сортировка вставками
- 2.3.5. Сортировка слиянием (*Merge_Sort*)
- 2.3.6. Быстрая сортировка (*Quick_Sort*)

2.4. Алгоритмы с применением деревьев
- 2.4.1. Концепция дерева
- 2.4.2. Бинарные деревья
- 2.4.3. Обходы деревьев
- 2.4.4. Представление выражений
- 2.4.5. Упорядоченные бинарные деревья
- 2.4.6. Сбалансированные бинарные деревья

2.5. Алгоритмы с применением *кучей*
- 2.5.1. Что такое *кучи*
- 2.5.2. Алгоритм *сортировки кучей*
- 2.5.3. Очереди с приоритетом

2.6. Алгоритмы на графах
- 2.6.1. Представление
- 2.6.2. Обход в ширину
- 2.6.3. Обход в глубину
- 2.6.4. Топологическая сортировка

2.7. *Жадные* алгоритмы
- 2.7.1. *Жадная* стратегия
- 2.7.2. Элементы *жадной стратегии*
- 2.7.3. Обмен монет
- 2.7.4. Задача коммивояжера
- 2.7.5. Задача о рюкзаке

2.8. Поиск кратчайших путей
- 2.8.1. Задача о кратчайшем пути
- 2.8.2. Отрицательные дуги и циклы
- 2.8.3. Алгоритм Дейкстры

2.9. *Жадные* алгоритмы на графах
- 2.9.1. Минимальное остовное дерево
- 2.9.2. Алгоритм Прима
- 2.9.3. Алгоритм Краскала
- 2.9.4. Анализ сложности

2.10. *Техника Backtracking*
- 2.10.1. Техника *Backtracking*
- 2.10.2. Альтернативные техники

Модуль 3. Объектно-ориентированное программирование

3.1. Введение в объектно-ориентированное программирование
- 3.1.1. Введение в объектно-ориентированное программирование
- 3.1.2. Разработка классов
- 3.1.3. Введение в унифицированный язык моделирования (UML) для моделирования задач

3.2. Отношения между классами
- 3.2.1. Абстракция и наследование
- 3.2.2. Расширенные концепции наследования
- 3.2.3. Полиморфизм
- 3.2.4. Состав и агрегация

3.3. Введение в паттерны проектирования для объектно-ориентированных задач
- 3.3.1. Что такое паттерны проектирования?
- 3.3.2. Паттерн Фабрика
- 3.3.4. Паттерн Одиночка
- 3.3.5. Паттерн Наблюдатель
- 3.3.6. Паттерн Компоновщик

3.4. Исключения
- 3.4.1. Что такое исключения?
- 3.4.2. Перехват и обработка исключений
- 3.4.3. Запуск исключений
- 3.4.4. Создание исключений

3.5. Пользовательские интерфейсы
- 3.5.1. Введение во фреймворк Qt
- 3.5.2. Позиционирование
- 3.5.3. Что такое события?
- 3.5.4. События: определение и захват
- 3.5.5. Разработка пользовательского интерфейса

3.6. Введение в параллельное программирование
- 3.6.1. Введение в параллельное программирование
- 3.6.2. Концепция процесса и потока
- 3.6.3. Взаимодействие между процессами или потоками
- 3.6.4. Потоки в C++
- 3.6.5. Преимущества и недостатки параллельного программирования

3.7. Управление потоками и синхронизация
- 3.7.1. Жизненный цикл потока
- 3.7.2. Класс *Thread*
- 3.7.3. Планирование потоков
- 3.7.4. Группы потоков
- 3.7.5. Демонические потоки
- 3.7.6. Синхронизация
- 3.7.7. Механизмы блокировки
- 3.7.8. Механизмы коммуникации
- 3.7.9. Мониторы

3.8. Распространенные задачи в параллельном программировании
- 3.8.1. Задача "производитель – потребитель"
- 3.8.2. Задача "читатель-писатель"
- 3.8.3. Задача об обедающих философах

3.9. Документация и тестирование ПО
- 3.9.1. Почему важно документировать ПО?
- 3.9.2. Проектный документ
- 3.9.3. Использование инструментов для документирования

3.10. Тестирование ПО
- 3.10.1. Введение в тестирование ПО
- 3.10.2. Виды тестирования
- 3.10.3. Единичное тестирование
- 3.10.4. Интеграционное тестирование
- 3.10.5. Валидационное тестирование
- 3.10.6. Тестирование системы

Модуль 4. Консоли и игровые устройства

4.1. История программирования видеоигр
- 4.1.1. Период Atari (1977–1985)
- 4.1.2. Период NES и SNES (1985–1995)
- 4.1.3. Период PlayStation / PlayStation 2 (1995–2005)
- 4.1.4. Период Xbox 360, PS3 и Wii (2005–2013)
- 4.1.5. Период Xbox One, PS4 и Wii U-Switch (с 2013 года по настоящее время)
- 4.1.6. Будущее

4.2. История геймплея в видеоиграх
- 4.2.1. Введение
- 4.2.2. Социальный контекст
- 4.2.3. Структурная диаграмма
- 4.2.4. Будущее

4.3. Адаптация к современности
- 4.3.1. Игры, основанные на движении
- 4.3.2. Виртуальная реальность
- 4.3.3. Дополненная реальность
- 4.3.4. Смешанная реальность

4.4. *Unity: Скриптинг* I и примеры
- 4.4.1. Что такое *скрипт*?
- 4.4.2. Наш первый *скрипт*
- 4.4.3. Добавление *скрипта*
- 4.4.4. Открытие *скрипта*
- 4.4.5. MonoBehaviour
- 4.4.6. *Debugging*

4.5. *Unity: Скриптинг* II и примеры
- 4.5.1. Ввод с клавиатуры и мыши
- 4.5.2. Raycast
- 4.5.3. Инстанцирование
- 4.5.4. Переменные
- 4.5.5. Открытые и сериализованные переменные

4.6. *Unity: Скриптинг* III и примеры
- 4.6.1. Получаем компоненты
- 4.6.2. Модифицируем компоненты
- 4.6.3. Тестирование
- 4.6.4. Множественные объекты
- 4.6.5. Колайдеры и триггеры
- 4.6.6. Кватернионы

4.7. Периферийные устройства
- 4.7.1. Эволюция и классификация
- 4.7.2. Периферийные устройства и интерфейсы
- 4.7.3. Современные периферийные устройства
- 4.7.4. Ближайшее будущее

4.8. Видеоигры: перспективы на будущее
- 4.8.1. Облачные игры
- 4.8.2. Отсутствие контроллеров
- 4.8.3. Иммерсивная реальность
- 4.8.4. Другие альтернативы

4.9. Архитектура
- 4.9.1. Специфические потребности видеоигр
- 4.9.2. Эволюция архитектуры
- 4.9.3. Современная архитектура
- 4.9.4. Различия между архитектурами

4.10. Средства разработки и их эволюция
- 4.10.1. Введение
- 4.10.2. Средства разработки третьего поколения
- 4.10.3. Средства разработки четвертого поколения
- 4.10.4. Средства разработки пятого поколения
- 4.10.5. Средства разработки шестого поколения

Модуль 5. Программная инженерия

5.1. Введение в разработку ПО и моделирование
- 5.1.1. Природа ПО
- 5.1.2. Уникальная природа веб-приложений
- 5.1.3. Разработка ПО
- 5.1.4. Процесс разработки ПО
- 5.1.5. Практика разработки ПО
- 5.1.6. Мифы о ПО
- 5.1.7. Как все начинается
- 5.1.8. Объектно-ориентированные концепции
- 5.1.9. Введение в UML

5.2. Процесс разработки ПО
- 5.2.1. Общая модель процесса
- 5.2.2. Предписывающие модели процессов
- 5.2.3. Специализированные модели процессов
- 5.2.4. Унифицированный процесс
- 5.2.5. Модели персональных и командных процессов
- 5.2.6. Что такое гибкая разработка?
- 5.2.7. Что такое Agile-процесс?
- 5.2.8. Scrum
- 5.2.9. Набор инструментов Agile-процессов

5.3. Принципы, определяющие практику разработки ПО
- 5.3.1. Принципы, направляющие процесс
- 5.3.2. Принципы, направляющие практику
- 5.3.3. Принципы коммуникации
- 5.3.4. Принципы планирования
- 5.3.5. Принципы моделирования
- 5.3.6. Принципы построения
- 5.3.7. Принципы развертывания

5.4. Понимание требований
- 5.4.1. Разработка требований
- 5.4.2. Закладывание основ
- 5.4.3. Исследование требований
- 5.4.4. Разработка сценариев использования
- 5.4.5. Разработка модели требований
- 5.4.6. Согласование требований
- 5.4.7. Валидация требований

5.5. Моделирование требований: Сценарии, информация и виды анализа
- 5.5.1. Анализ требований
- 5.5.2. Моделирование на основе сценариев
- 5.5.3. UML-модели, предоставляющие сценарий использования
- 5.5.4. Концепции моделирования данных
- 5.5.5. Моделирование на основе классов
- 5.5.6. Диаграммы классов

5.6. Моделирование требований: Поток, поведение и паттерны
- 5.6.1. Стратегии формирования требований
- 5.6.2. Моделирование, ориентированное на поток
- 5.6.3. Диаграммы состояний
- 5.6.4. Создание поведенческой модели
- 5.6.5. Диаграммы последовательностей
- 5.6.6. Коммуникационные диаграммы
- 5.6.7. Паттерны для моделирования требований

5.7. Концепции разработки
- 5.7.1. Проектирование в контексте разработки ПО
- 5.7.2. Процесс разработки
- 5.7.3. Концепции разработки
- 5.7.4. Концепции объектно-ориентированной разработки
- 5.7.5. Проектная модель

5.8. Архитектурное проектирование
- 5.8.1. Архитектура ПО
- 5.8.2. Архитектурные жанры
- 5.8.3. Архитектурные стили
- 5.8.4. Архитектурное проектирование
- 5.8.5. Эволюция альтернативных проектов архитектуры
- 5.8.6. Составление карты архитектуры с использованием потоков данных

5.9. Разработка на компонентном уровне и на основе паттернов
- 5.9.1. Что такое компонент?
- 5.9.2. Разработка компонентов на основе классов
- 5.9.3. Реализация проекта на уровне компонентов
- 5.9.4. Разработка традиционных компонентов
- 5.9.5. Разработка на основе компонентов
- 5.9.6. Паттерны проектирования
- 5.9.7. Разработка ПО на основе паттернов
- 5.9.8. Архитектурные паттерны
- 5.9.9. Паттерны проектирования на уровне компонентов
- 5.9.10. Паттерны разработки пользовательского интерфейса

5.10. Качество ПО и управление проектами
- 5.10.1. Качество
- 5.10.2. Качество программного обеспечения
- 5.10.3. Дилемма качества ПО
- 5.10.4. Достижение качества ПО
- 5.10.5. Обеспечение качества ПО
- 5.10.6. Управленческий спектр
- 5.10.7. Персонал
- 5.10.8. Продукт
- 5.10.9. Процесс
- 5.10.10. Проект
- 5.10.11. Принципы и практика

Модуль 6. Движки для разработки игр

6.1. Видеоигры и ИКТ
- 6.1.1. Введение
- 6.1.2. Возможности
- 6.1.3. Вызовы
- 6.1.4. Выводы

6.2. История движков для видеоигр
- 6.2.1. Введение
- 6.2.2. Atari
- 6.2.3. 1980-е годы
- 6.2.4. Первые движки. 1990-е годы
- 6.2.5. Современные движки

6.3. Движки для разработки игр
- 6.3.1. Виды движков
- 6.3.2. Части движка видеоигры
- 6.3.3. Современные движки
- 6.3.4. Выбор движка для нашего проекта

6.4. *Движок Game Maker*
- 6.4.1. Введение
- 6.4.2. Разработка сценария
- 6.4.3. Спрайты и анимации
- 6.4.4. Столкновения
- 6.4.5. *Скриптинг* в GML

6.5. Движок Unreal Engine 4: введение
- 6.5.1. Что такое Unreal Engine 4? В чем заключается его философия?
- 6.5.2. Материалы
- 6.5.3. UI
- 6.5.4. Анимация
- 6.5.5. Система частиц
- 6.5.6. Искусственный интеллект
- 6.5.7. FPS

6.6. Движок Unreal Engine 4: *Визуальное программирование*
- 6.6.1. Философия *блюпринта* и *визуального скриптинга*
- 6.6.2. *Debugging*
- 6.6.3. Типы переменных
- 6.6.4. Базовый контроль потока

6.7. Движок Unity 5
- 6.7.1. Программирование на C# и Visual Studio
- 6.7.2. Создание префабов
- 6.7.3. Использование гизмосов для управления видеоиграми
- 6.7.4. Адаптивный движок: 2D и 3D

6.8. Движок Godot
- 6.8.1. Философия дизайна Godot
- 6.8.2. Объектно-ориентированное проектирование и композиция
- 6.8.3. Все в одном пакете
- 6.8.4. Бесплатное и управляемое сообществом ПО

6.9. Движок RPG Maker
- 6.9.1. Философия RPG Maker
- 6.9.2. Взять за основу
- 6.9.3. Создание игры с индивидуальным характером
- 6.9.4. Успешные коммерческие игры

6.10. Движок Source 2
- 6.10.1. Философия Source 2
- 6.10.2. Source и Source 2: развитие
- 6.10.3. Общественное использование: Аудиовизуальные материалы и видеоигры
- 6.10.4. Будущее движка Source 2
- 6.10.5. Моды и успешные игры

Модуль 7. Интеллектуальные системы

7.1. Теория агентов
- 7.1.1. История концепции
- 7.1.2. Определение агента
- 7.1.3. Агенты в системах искусственного интеллекта
- 7.1.4. Агенты в разработке программного обеспечения

7.2. Архитектуры агентов
- 7.2.1. Процесс рассуждения агента
- 7.2.2. Реактивные агенты
- 7.2.3. Дедуктивные агенты
- 7.2.4. Гибридные агенты
- 7.2.5. Сравнение

7.3. Информация и знания
- 7.3.1. Различие между данными, информацией и знаниями
- 7.3.2. Оценка качества данных
- 7.3.3. Методы сбора данных
- 7.3.4. Методы получения информации
- 7.3.5. Методы приобретения знаний

7.4. Представление знаний
- 7.4.1. Важность представления знаний
- 7.4.2. Определение представления знаний через их роли
- 7.4.3. Характеристики представления знаний

7.5. Онтологии
- 7.5.1. Введение в метаданные
- 7.5.2. Философская концепция онтологии
- 7.5.3. Вычислительная концепция онтологии
- 7.5.4. Онтологии доменов и онтологии более высокого уровня
- 7.5.5. Как создать онтологию

7.6. Языки онтологий и ПО для создания онтологий
- 7.6.1. Семантическая тройка RDF, Turtle и N3
- 7.6.2. Схема RDF
- 7.6.3. OWL
- 7.6.4. SPARQL
- 7.6.5. Знакомство с различными инструментами для создания онтологий
- 7.6.6. Установка и использование Protégé

7.7. Семантическая паутина
- 7.7.1. Текущее состояние и будущее семантической паутины
- 7.7.2. Семантические веб-приложения

7.8. Другие модели представления знаний
- 7.8.1. Словари
- 7.8.2. Глобальный обзор
- 7.8.3. Таксономия
- 7.8.4. Тезаурусы
- 7.8.5. Фолксономии
- 7.8.6. Сравнение
- 7.8.7. Ментальные карты

7.9. Оценка и интеграция представлений знаний
- 7.9.1. Логика нулевого порядка
- 7.9.2. Логика первого порядка
- 7.9.3. Дескрипционная логика
- 7.9.4. Взаимосвязь между различными типами логики
- 7.9.5. Пролог: Программирование на основе логики первого порядка

7.10. Семантические анализаторы, системы, основанные на знаниях, и экспертные системы
- 7.10.1. Концепция анализатора
- 7.10.2. Применение анализатора
- 7.10.3. Системы, основанные на знаниях
- 7.10.4. MYCIN, история экспертных систем
- 7.10.5. Элементы и архитектура экспертных систем
- 7.10.6. Создание экспертных систем

Модуль 8. Программирование в реальном времени

8.1. Основы параллельного программирования
- 8.1.1. Фундаментальные концепции
- 8.1.2. Параллелизм
- 8.1.3. Преимущества параллелизма
- 8.1.4. Параллелизм и аппаратное обеспечение

8.2. Основные структуры поддержки параллелизма в Java
- 8.2.1. Параллелизм в Java
- 8.2.2. Создание *потоков*
- 8.2.3. Методы
- 8.2.4. Синхронизация

8.3. *Потоки*, жизненный цикл, приоритеты, прерывания, состояния, исполнители
- 8.3.1. *Потоки*
- 8.3.2. Жизненный цикл
- 8.3.3. Приоритеты
- 8.3.4. Прерывания
- 8.3.5. Состояния
- 8.3.6. Исполнители

8.4. Взаимное исключение
- 8.4.1. Что такое взаимное исключение?
- 8.4.2. Алгоритм Деккера
- 8.4.3. Алгоритм Петерсона
- 8.4.4. Взаимное исключение в Java

8.5. Зависимости состояний
- 8.5.1. Внедрение зависимостей
- 8.5.2. Реализация паттерна в Java
- 8.5.3. Способы внедрения зависимостей
- 8.5.4. Пример

8.6. Паттерны проектирования
- 8.6.1. Введение
- 8.6.2. Порождающие паттерны
- 8.6.3. Структурные паттерны
- 8.6.4. Поведенческие паттерны

8.7. Использование библиотек Java
- 8.7.1. Что такое библиотеки в Java?
- 8.7.2. Mockito-All, Mockito-Core
- 8.7.3. Guava
- 8.7.4. Commons-io
- 8.7.5. Commons-lang, commons-Lang3

8.8. Программирование шейдеров
- 8.8.1. Пайплайн в 3D и создание растрового изображения
- 8.8.2. Вертексное затенение
- 8.8.3. Пиксельное затенение: Освещение I
- 8.8.4. Пиксельное затенение: Освещение II
- 8.8.5. Пост-эффекты

8.9. Программирование в реальном времени
- 8.9.1. Введение
- 8.9.2. Обработка прерываний
- 8.9.3. Синхронизация и коммуникация между процессами
- 8.9.4. Системы планирования в реальном времени

8.10. Планирование в реальном времени
- 8.10.1. Концепции
- 8.10.2. Эталонная модель для систем реального времени
- 8.10.3. Политики планирования
- 8.10.4. Циклические планировщики
- 8.10.5. Планировщики со статическими свойствами
- 8.10.6. Планировщики с динамическими свойствами

Модуль 9. Проектирование и разработка веб-игр

9.1. Происхождение и стандарты веб
- 9.1.1. Происхождение сети Интернет
- 9.1.2. Создание Всемирной паутины
- 9.1.3. Появление веб-стандартов
- 9.1.4. Популярность веб-стандартов

9.2. HTTP и структура "клиент — сервер"
- 9.2.1. Роли клиента и сервера
- 9.2.2. Коммуникация клиента и сервера
- 9.2.3. Новейшая история
- 9.2.4. Централизованные вычисления

9.3. Веб-программирование: Введение
- 9.3.1. Основные понятия
- 9.3.2. Подготовка веб-сервера
- 9.3.3. Основные концепции HTML5
- 9.3.4. HTML-формы

9.4. Введение в HTML и примеры
- 9.4.1. История HTML5
- 9.4.2. Элементы HTML5
- 9.4.3. APIs
- 9.4.4. CCS3

9.5. Объектная модель документа
- 9.5.1. Что такое объектная модель документа?
- 9.5.2. Использование DOCTYPE
- 9.5.3. Важность валидации HTML
- 9.5.4. Доступ к элементам
- 9.5.5. Создание элементов и текстов
- 9.5.6. Использование innerHTML
- 9.5.7. Удаление текстового элемента или узла
- 9.5.8. Чтение и запись атрибутов элемента
- 9.5.9. Манипулирование стилями элементов
- 9.5.10. Прикрепление нескольких файлов одновременно

9.6. Введение в CSS и примеры
- 9.6.1. Синтаксис CSS3
- 9.6.2. Таблицы стилей
- 9.6.3. Ярлыки
- 9.6.4. Селекторы
- 9.6.5. Веб-разработка с использованием CSS

9.7. Введение в JavaScript и примеры
- 9.7.1. Что такое JavaScript?
- 9.7.2. Краткая история языка
- 9.7.3. Версии JavaScript
- 9.7.4. Отображение диалогового окна
- 9.7.5. Синтаксис JavaScript
- 9.7.6. Понимание *скриптов*
- 9.7.7. План пространства
- 9.7.8. Комментарии
- 9.7.9. Функции
- 9.7.10. Внутренний и внешний JavaScript

9.8. Функции в JavaScript
- 9.8.1. Функциональные утверждения
- 9.8.2. Функциональные выражения
- 9.8.3. Функции вызова
- 9.8.4. Рекурсия
- 9.8.5. Вложенные функции и закрытия
- 9.8.6. Сохранение переменных
- 9.8.7. Мультивложенные функции
- 9.8.8. Конфликты имен
- 9.8.9. Закрытия или замыкания
- 9.8.10. Параметры функции

9.9. PlayCanvas для разработки веб-игр
- 9.9.1. Что такое PlayCanvas?
- 9.9.2. Конфигурация проекта
- 9.9.3. Создание объекта
- 9.9.4. Добавление физики
- 9.9.5. Добавление модели
- 9.9.6. Изменение параметров гравитации и сцены
- 9.9.7. Выполнение *скриптов*
- 9.9.8. Контролеры камеры

9.10. Phaser для разработки веб-игр
- 9.10.1. Что такое Phaser?
- 9.10.2. Загрузка ресурсов
- 9.10.3. Построение мира
- 9.10.4. Платформы
- 9.10.5. Игрок
- 9.10.6. Добавление физики
- 9.10.7. Использование клавиатуры
- 9.10.8. Подбор *пикапов*
- 9.10.9. Очки и подсчет очков
- 9.10.10. Прыгающая бомба

Модуль 10. Многопользовательские сети и системы

10.1. История и эволюция многопользовательских видеоигр
- 10.1.1. 1970-е Первые многопользовательские игры
- 10.1.2. 90-е годы: Duke Nukem, Doom, Quake
- 10.1.3. Расцвет многопользовательских видеоигр
- 10.1.4. Локальный и сетевой мультиплеер
- 10.1.5. Игры для вечеринок

10.2. Многопользовательские бизнес-модели
- 10.2.1. Происхождение и функционирование возникающих бизнес-моделей
- 10.2.2. Сервисы онлайн-продаж
- 10.2.3. Играть бесплатно
- 10.2.4. Микроплатежи
- 10.2.5. Реклама
- 10.2.6. Подписка с ежемесячными платежами
- 10.2.7. Заплатить, чтобы поиграть
- 10.2.8. Попробовать, прежде чем купить

10.3. Локальные и сетевые игры
- 10.3.1. Локальные игры: зарождение
- 10.3.2. Игры для вечеринок: Nintendo и единение семьи
- 10.3.3. Сетевые игры: зарождение
- 10.3.4. Развитие сетевых игр

10.4. Модель OSI: Уровни I
- 10.4.1. Модель OSI: введение
- 10.4.2. Физический уровень
- 10.4.3. Канальный уровень
- 10.4.4. Сетевой уровень

10.5. Модель OSI: Уровни II
- 10.5.1. Транспортный уровень
- 10.5.2. Сеансовый уровень
- 10.5.3. Уровень представления
- 10.5.4. Прикладной уровень

10.6. Компьютерные сети и интернет
- 10.6.1. Что такое компьютерная сеть?
- 10.6.2. Программное обеспечение
- 10.6.3. Аппаратное обеспечение
- 10.6.4. Серверы
- 10.6.5. Сетевое хранилище
- 10.6.6. Сетевые протоколы

10.7. Мобильные и беспроводные сети
- 10.7.1. Мобильная сеть
- 10.7.2. Беспроводная сеть
- 10.7.3. Эксплуатация мобильных сетей
- 10.7.4. Цифровые технологии

10.8. Безопасность
- 10.8.1. Персональная безопасность
- 10.8.2. Взломы и читы в видеоиграх
- 10.8.3. Защита от взломов
- 10.8.4. Анализ систем против взломов

10.9. Многопользовательские системы: Серверы
- 10.9.1. Хостинг серверов
- 10.9.2. Массовые многопользовательские ролевые онлайн-игры
- 10.9.3. Выделенные серверы для видеоигр
- 10.9.4. LAN Parties

10.10. Дизайн и программирование многопользовательских видеоигр
- 10.10.1. Основы проектирования многопользовательских видеоигр в Unreal
- 10.10.2. Основы проектирования многопользовательских игр в Unity
- 10.10.3. Как сделать многопользовательскую игру увлекательной?
- 10.10.4. За рамками команды: Инновации в управлении мультиплеером

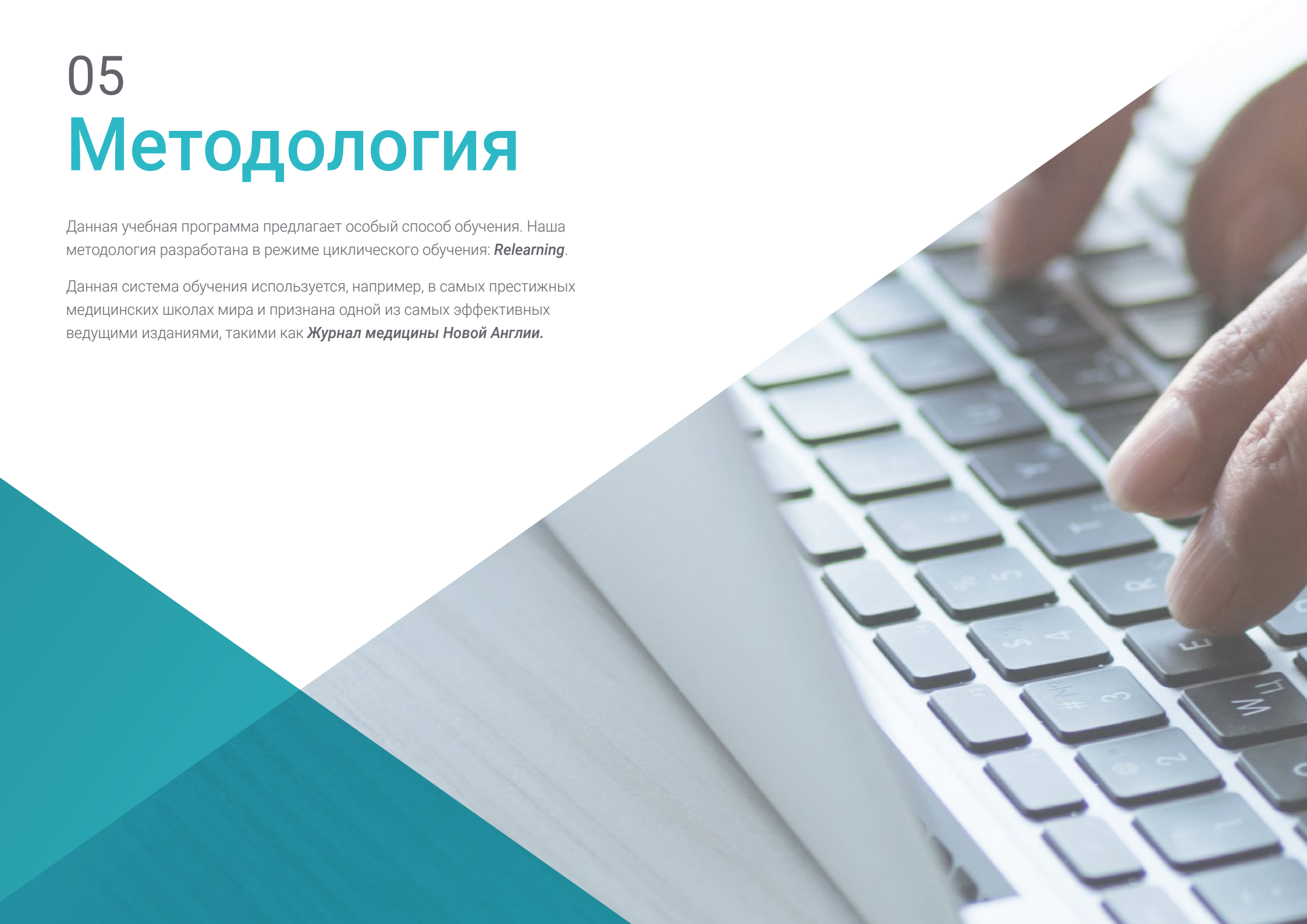
Если вы хотите сделать большую карьеру, программируя всемирно известные видеоигры, эта программа для вас"

05

Методология

Данная учебная программа предлагает особый способ обучения. Наша методология разработана в режиме циклического обучения: ***Relearning***.

Данная система обучения используется, например, в самых престижных медицинских школах мира и признана одной из самых эффективных ведущими изданиями, такими как ***Журнал медицины Новой Англии.***



Откройте для себя методику Relearning, которая отвергает традиционное линейное обучение, чтобы показать вам циклические системы обучения: способ, который доказал свою огромную эффективность, особенно в предметах, требующих запоминания"

Исследование кейсов для контекстуализации всего содержания

Наша программа предлагает революционный метод развития навыков и знаний. Наша цель - укрепить компетенции в условиях меняющейся среды, конкуренции и высоких требований.

С TECH вы сможете познакомиться со способом обучения, который опровергает основы традиционных методов образования в университетах по всему миру”

Вы получите доступ к системе обучения, основанной на повторении, с естественным и прогрессивным обучением по всему учебному плану.

В ходе совместной деятельности и рассмотрения реальных кейсов студент научится разрешать сложные ситуации в реальной бизнес-среде.

Инновационный и отличный от других метод обучения

Эта программа TECH - интенсивная программа обучения, созданная с нуля, которая предлагает самые сложные задачи и решения в этой области на международном уровне. Благодаря этой методологии ускоряется личностный и профессиональный рост, делая решающий шаг на пути к успеху. Метод кейсов, составляющий основу данного содержания, обеспечивает следование самым современным экономическим, социальным и профессиональным реалиям.

> *Наша программа готовит вас к решению новых задач в условиях неопределенности и достижению успеха в карьере"*

Кейс-метод является наиболее широко используемой системой обучения лучшими преподавателями в мире. Разработанный в 1912 году для того, чтобы студенты-юристы могли изучать право не только на основе теоретического содержания, метод кейсов заключается в том, что им представляются реальные сложные ситуации для принятия обоснованных решений и ценностных суждений о том, как их разрешить. В 1924 году он был установлен в качестве стандартного метода обучения в Гарвардском университете.

Что должен делать профессионал в определенной ситуации? Именно с этим вопросом мы сталкиваемся при использовании кейс-метода - метода обучения, ориентированного на действие. На протяжении всей курса студенты будут сталкиваться с многочисленными реальными случаями из жизни. Им придется интегрировать все свои знания, исследовать, аргументировать и защищать свои идеи и решения.

Методология *Relearning*

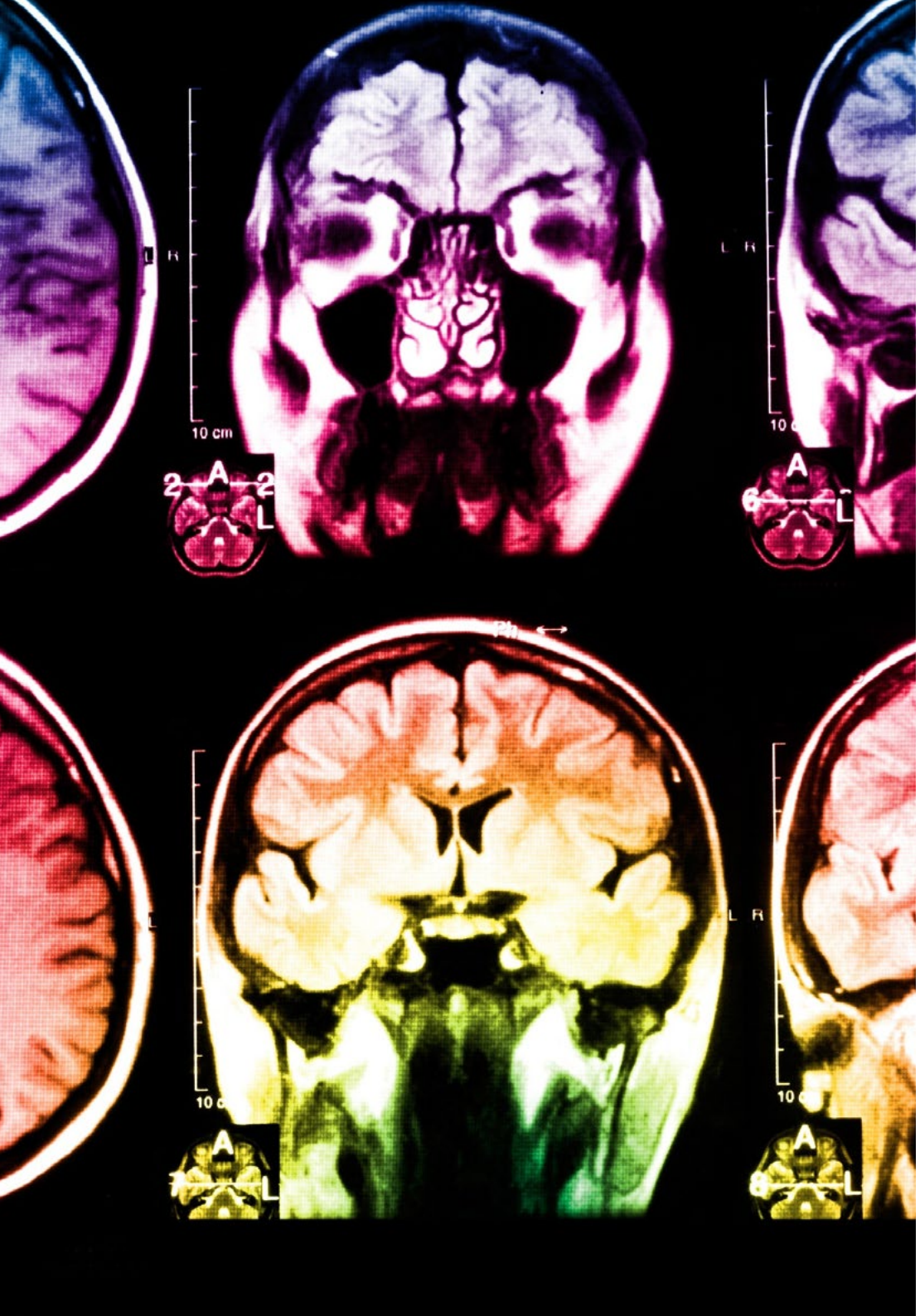
TECH эффективно объединяет метод кейсов с системой 100% онлайн-обучения, основанной на повторении, которая сочетает различные дидактические элементы в каждом уроке.

Мы улучшаем метод кейсов с помощью лучшего метода 100% онлайн-обучения: *Relearning*.

В 2019 году мы достигли лучших результатов обучения среди всех онлайн-университетов в мире.

В TECH вы будете учиться по передовой методике, разработанной для подготовки руководителей будущего. Этот метод, играющий ведущую роль в мировой педагогике, называется *Relearning*.

Наш университет - единственный вуз, имеющий лицензию на использование этого успешного метода. В 2019 году нам удалось повысить общий уровень удовлетворенности наших студентов (качество преподавания, качество материалов, структура курса, цели...) по отношению к показателям лучшего онлайн-университета.



В нашей программе обучение не является линейным процессом, а происходит по спирали (мы учимся, разучиваемся, забываем и заново учимся). Поэтому мы дополняем каждый из этих элементов по концентрическому принципу. Благодаря этой методике более 650 000 выпускников университетов добились беспрецедентного успеха в таких разных областях, как биохимия, генетика, хирургия, международное право, управленческие навыки, спортивная наука, философия, право, инженерное дело, журналистика, история, финансовые рынки и инструменты. Наша методология преподавания разработана в среде с высокими требованиями к уровню подготовки, с университетским контингентом студентов с высоким социально-экономическим уровнем и средним возрастом 43,5 года.

Методика Relearning позволит вам учиться с меньшими усилиями и большей эффективностью, все больше вовлекая вас в процесс обучения, развивая критическое мышление, отстаивая аргументы и противопоставляя мнения, что непосредственно приведет к успеху.

Согласно последним научным данным в области нейронауки, мы не только знаем, как организовать информацию, идеи, образы и воспоминания, но и знаем, что место и контекст, в котором мы что-то узнали, имеют фундаментальное значение для нашей способности запомнить это и сохранить в гиппокампе, чтобы удержать в долгосрочной памяти.

Таким образом, в рамках так называемого нейрокогнитивного контекстно-зависимого электронного обучения, различные элементы нашей программы связаны с контекстом, в котором участник развивает свою профессиональную практику.

В рамках этой программы вы получаете доступ к лучшим учебным материалам, подготовленным специально для вас:

Учебный материал

Все дидактические материалы создаются преподавателями специально для студентов этого курса, чтобы они были действительно четко сформулированными и полезными.

Затем вся информация переводится в аудиовизуальный формат, создавая дистанционный рабочий метод TECH. Все это осуществляется с применением новейших технологий, обеспечивающих высокое качество каждого из представленных материалов.

Мастер-классы

Существуют научные данные о пользе экспертного наблюдения третьей стороны.

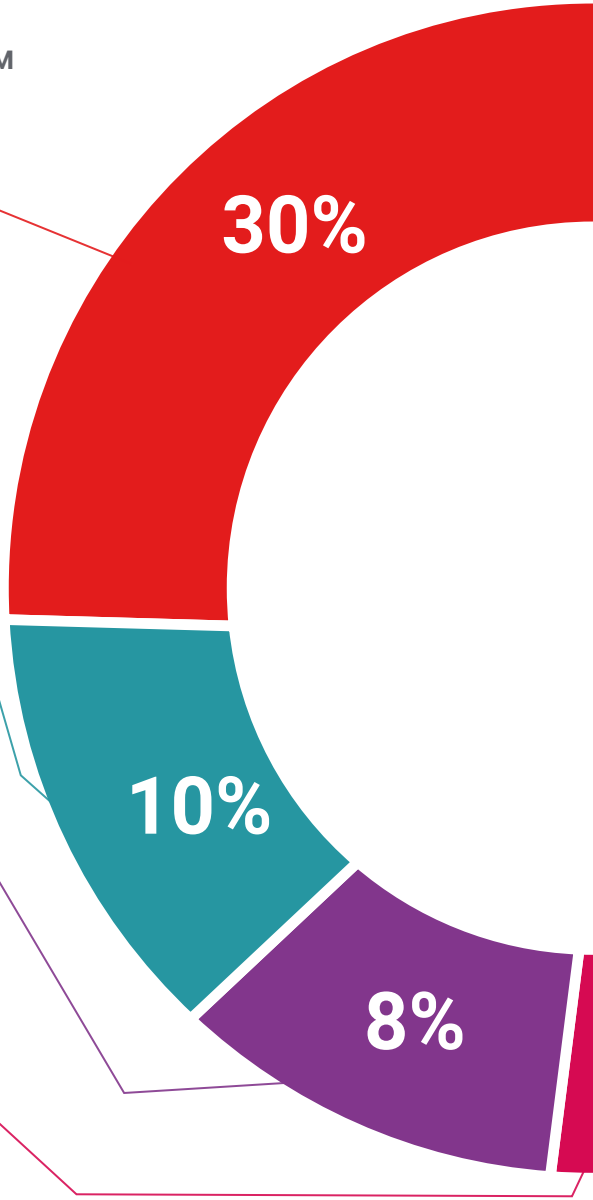
Так называемый метод обучения у эксперта укрепляет знания и память, а также формирует уверенность в наших будущих сложных решениях.

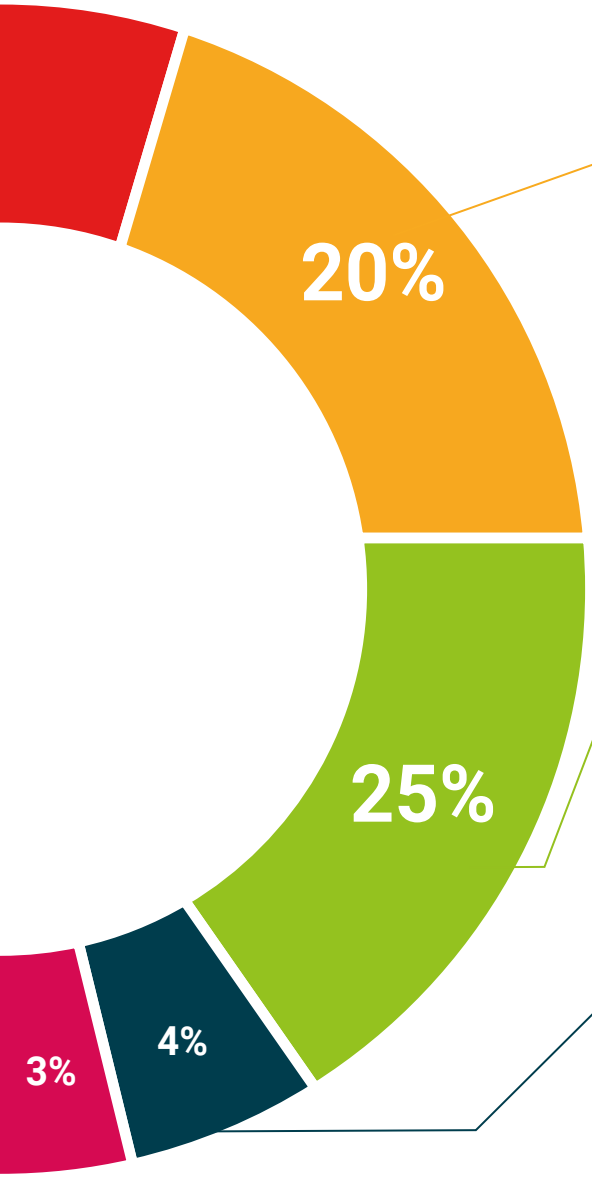
Практика навыков и компетенций

Студенты будут осуществлять деятельность по развитию конкретных компетенций и навыков в каждой предметной области. Практика и динамика приобретения и развития навыков и способностей, необходимых специалисту в рамках глобализации, в которой мы живем.

Дополнительная литература

Новейшие статьи, консенсусные документы и международные руководства включены в список литературы курса. В виртуальной библиотеке TECH студент будет иметь доступ ко всем материалам, необходимым для завершения обучения.





Метод кейсов

Метод дополнится подборкой лучших кейсов, выбранных специально для этой квалификации. Кейсы представляются, анализируются и преподаются лучшими специалистами на международной арене.

Интерактивные конспекты

Мы представляем содержание в привлекательной и динамичной мультимедийной форме, которая включает аудио, видео, изображения, диаграммы и концептуальные карты для закрепления знаний.

Эта уникальная обучающая система для представления мультимедийного содержания была отмечена компанией Microsoft как "Европейская история успеха".

Тестирование и повторное тестирование

На протяжении всей программы мы периодически оцениваем и переоцениваем ваши знания с помощью оценочных и самооценочных упражнений: так вы сможете убедиться, что достигаете поставленных целей.

06

Квалификация

Специализированная магистратура в области Программирование видеоигр гарантирует, помимо самого строгого и современного обучения, получение диплома об окончании Специализированной магистратуры, выдаваемого TECH Технологическим университетом.

Успешно пройдите эту программу и получите университетский диплом без хлопот, связанных с поездками и оформлением документов"

Данная **Специализированная магистратура в области Программирование видеоигр** содержит самую полную и современную программу на рынке.

После прохождения аттестации студент получит по почте* с подтверждением получения соответствующий диплом **Специализированной магистратуры,** выданный **ТЕСН Технологическим университетом.**

Диплом, выданный **ТЕСН Технологическим университетом**, подтверждает квалификацию, полученную в Специализированной магистратуре, и соответствует требованиям, обычно предъявляемым биржами труда, конкурсными экзаменами и комитетами по оценке карьеры.

Диплом: **Специализированная магистратура в области Программирование видеоигр**

Количество учебных часов: **1500 часов**


tech технологический университет

Настоящий

ДИПЛОМ

свидетельствует о том, что

Г-н/Г-жа _____, с документом, удостоверяющий личность_____

Освоил(а) и успешно прошел(ла) аккредитацию программы

СПЕЦИАЛИЗИРОВАННАЯ МАГИСТРАТУРА

в области

Программирование видеоигр

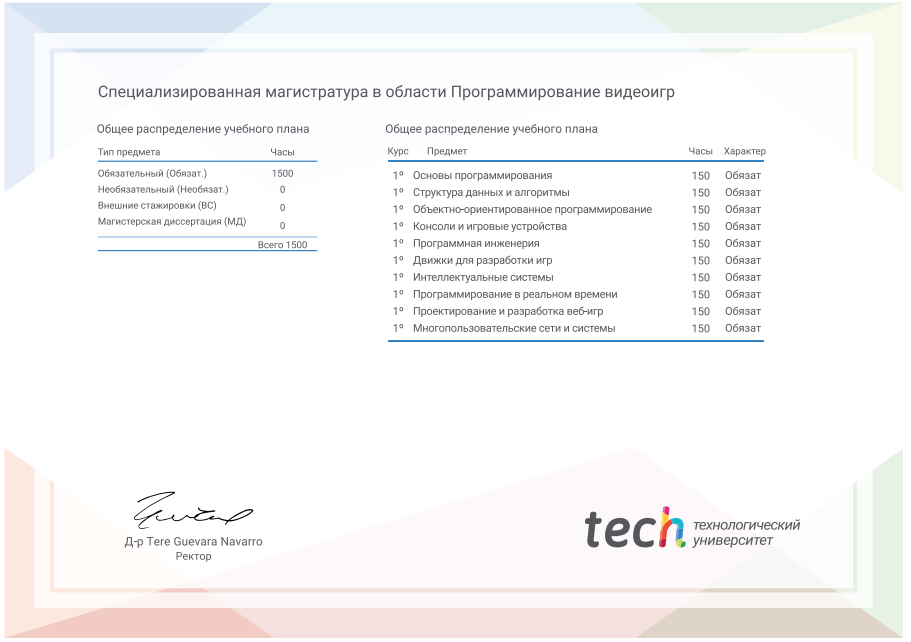
Данный диплом специализированной программы, присуждаемый Университетом, соответствует 1500 учебным часам, с датой начала дд/мм/гггг и датой окончания дд/мм/гггг.

ТЕСН является частным высшим учебным заведением, признанным Министерством народного образования Мексики с 28 июня 2018 года.

17 июня 2020 г.

Д-р Tere Guevara Navarro
Ректор

unique TECH code: AFWOR23S techtitute.com/certificates


Специализированная магистратура в области Программирование видеоигр

Общее распределение учебного плана

Тип предмета	Часы
Обязательный (Обязат.)	1500
Необязательный (Необязат.)	0
Внешние стажировки (ВС)	0
Магистерская диссертация (МД)	0
Всего 1500	

Общее распределение учебного плана

Курс	Предмет	Часы	Характер
1°	Основы программирования	150	Обязат
1°	Структура данных и алгоритмы	150	Обязат
1°	Объектно-ориентированное программирование	150	Обязат
1°	Консоли и игровые устройства	150	Обязат
1°	Программная инженерия	150	Обязат
1°	Движки для разработки игр	150	Обязат
1°	Интеллектуальные системы	150	Обязат
1°	Программирование в реальном времени	150	Обязат
1°	Проектирование и разработка веб-игр	150	Обязат
1°	Многопользовательские сети и системы	150	Обязат

Д-р Tere Guevara Navarro
Ректор

tech технологический университет

*Гаагский апостиль. В случае, если студент потребует, чтобы на его диплом в бумажном формате был проставлен Гаагский апостиль, TECH EDUCATION предпримет необходимые шаги для его получения за дополнительную плату.

tech технологический университет

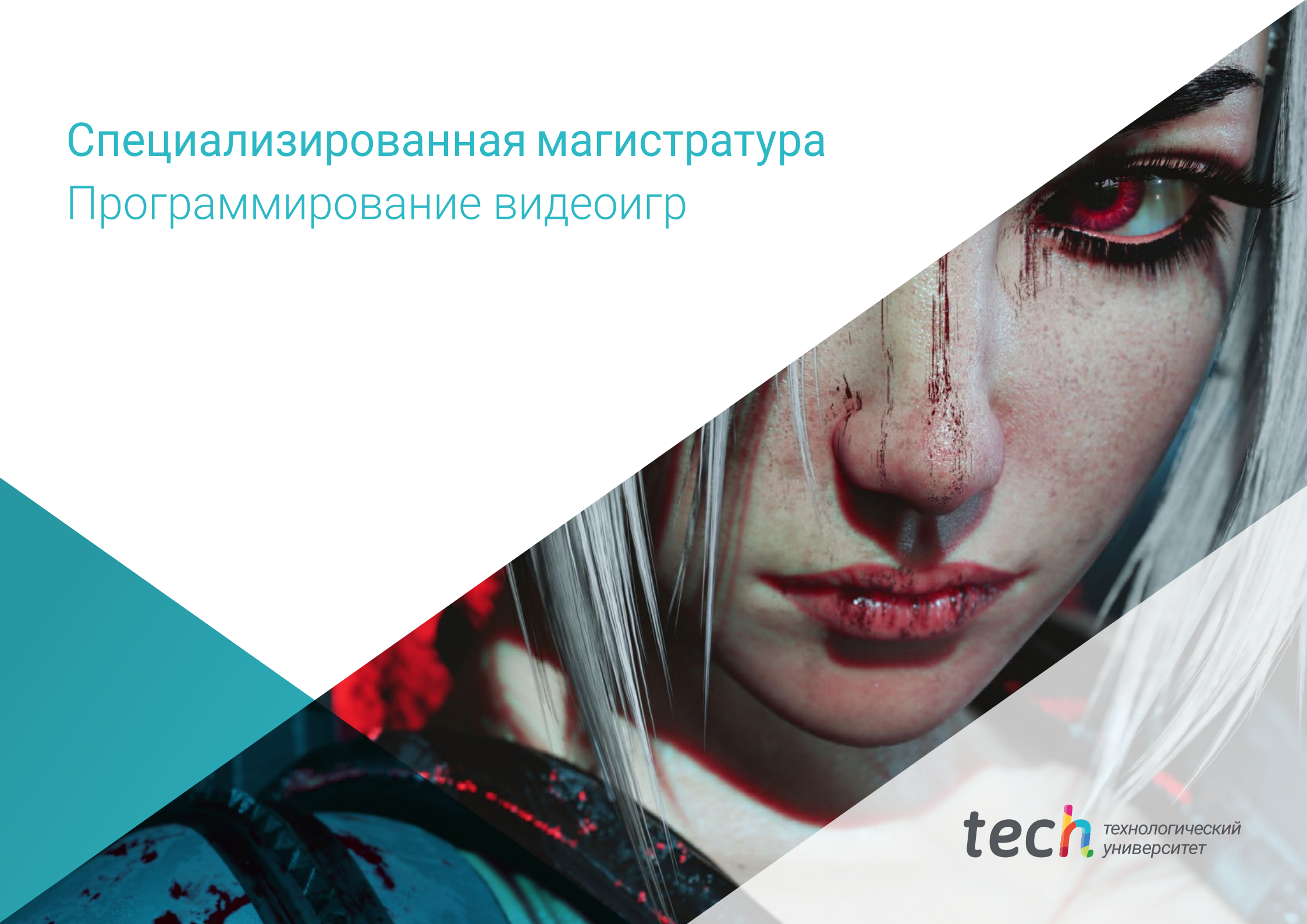
Специализированная магистратура

Программирование видеоигр

- Формат: **онлайн**
- Продолжительность: **12 месяцев**
- Учебное заведение: **TECH Технологический университет**
- Режим обучения: **16ч./неделя**
- Расписание: **по своему усмотрению**
- Экзамены: **онлайн**

Специализированная магистратура

Программирование видеоигр



tech технологический университет