校级硕士
视频游戏编程

tech 科学技术大学

tech 科学技术大学

校级硕士

视频游戏编程

- 模式:**在线**
- 时间:**12个月**
- 学历:**TECH科技大学**
- 时间:**16小时/周**
- 时间表:**按你方便的**
- 考试:**在线**

网络访问: www.techtitute.com/cn/information-technology/professional-master-degree/master-video-game-programming

目录

01

介绍

电子游戏最大的吸引力在于其最直观的方面，如其图形或设计。但如果没有程序设计，他们就无法脱颖而出。程序设计是任何电子游戏的关键，因为它决定了游戏的可玩性或图形如何与玩家互动。如果没有良好的程序设计，任何游戏都会失败，因为它将有无数的错误，也不会是一个令人愉快的体验。公司意识到这一点，这就是为什么他们需要高水平的开发人员。这个学位响应了这一需求，因为它为学生准备了能够面对该行业所有挑战的能力，所以他们会因为这个学位而获得众多的职业机会。

公司知道，视频游戏的关键在于程序设计。专业化并成为你所处环境中最受欢迎的开发者”

每一个伟大的电子游戏背后都有一个巨大的专业团队，他们专门负责每个领域的工作，努力使你的公司获得成功。通常情况下，对粉丝来说，最突出的部分是他们可以直接感知的部分，如视觉或与角色控制，机械或物体互动相关的部分。

然而，要使所有这些元素发挥作用并正确地整合，有一项基本的任务通常没有被考虑到：程序设计。视频游戏的开发有不同的阶段，涉及不同的部门，但程序设计是使一切有意义的，并形成基本的骨架，其他领域将被纳入。

这就是为什么该行业的公司如此重视这个问题，因为他们知道，正确和有效的电子游戏开发将促进项目的进展，避免出现错误和 Bugs.这就是为什么他们要寻找专门从事这一领域的最佳程序员。

但要找到该领域的真正专家并不容易。而这个视频游戏程序设计校级硕士正好满足了这一需求，使学生成为视频游戏开发方面的伟大专家，可以在这个行业中轻松发展，由于在这个学位上获得的技能和能力而获得巨大的职业机会。

这个**视频游戏编程的校级硕士**包含市场上最完整和最新的课程。主要特点是：

- 由视频游戏编程和开发的专家介绍案例研究的发展。
- 该书的内容图文并茂，示意性强，实用性强为那些视专业实践至关重要的学科提供了科学和实用的信息
- 实际练习，你可以进行自我评估过程，以改善你的学习
- 其特别强调创新方法
- 理论课，向专家提问，关于有争议问题的讨论区和个人反思性论文
- 可以从任何有互联网连接的固定或便携式设备上获取内容

> 由于这个校级硕士，在世界最好的公司开发所有类型的视频游戏”

在视频游戏的开发中，编程越来越重要。由于这个学位，成为该行业的一个重要组成部分”

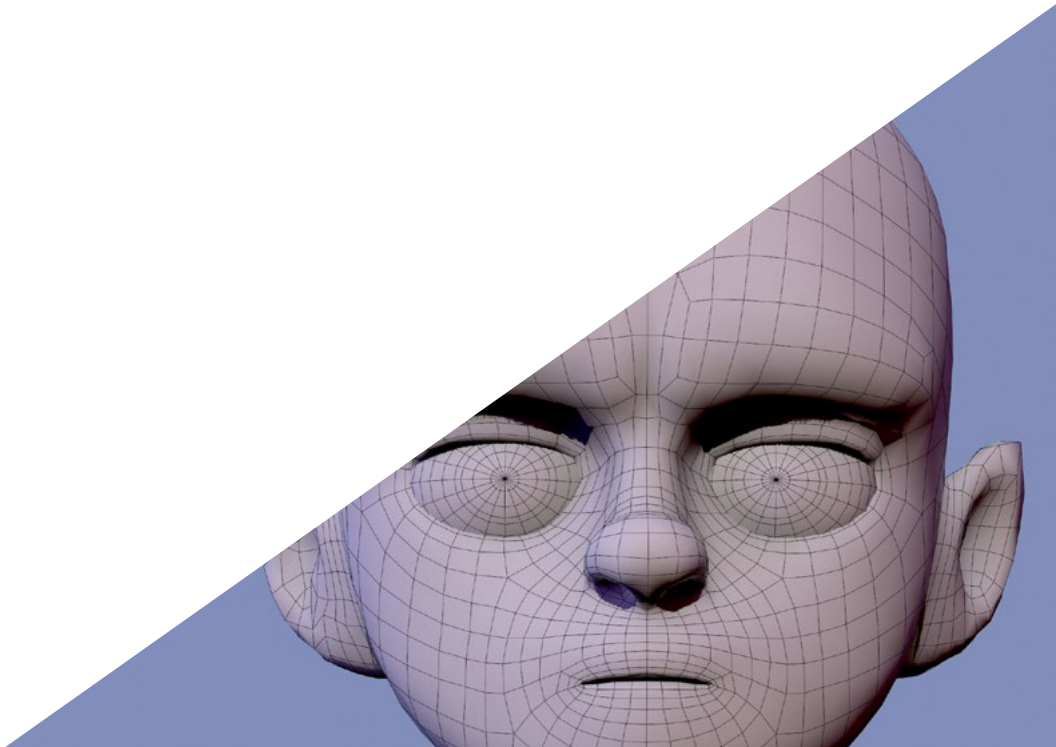
游戏是你的激情所在，你想成为一个伟大的开发者。不要再等了，赶紧报名参加这个校级硕士的学习。

行业内最好的公司在等着你。现在专攻。

该课程的教学人员包括来自该行业的专业人士，他们将自己的工作经验带到了这一培训中，还有来自领先公司和著名大学的公认专家。

多媒体内容是用最新的教育技术开发的，将允许专业人员进行情景式学习，即一个模拟的环境，提供一个身临其境的培训，为真实情况进行培训。

该课程的设计重点是**基于问题的学习**，通过这种方式，专业人员必须尝试解决整个学年出现的不同专业实践情况。它将得到一个由著名专家开发的创新互动视频系统的支持。



02

目标

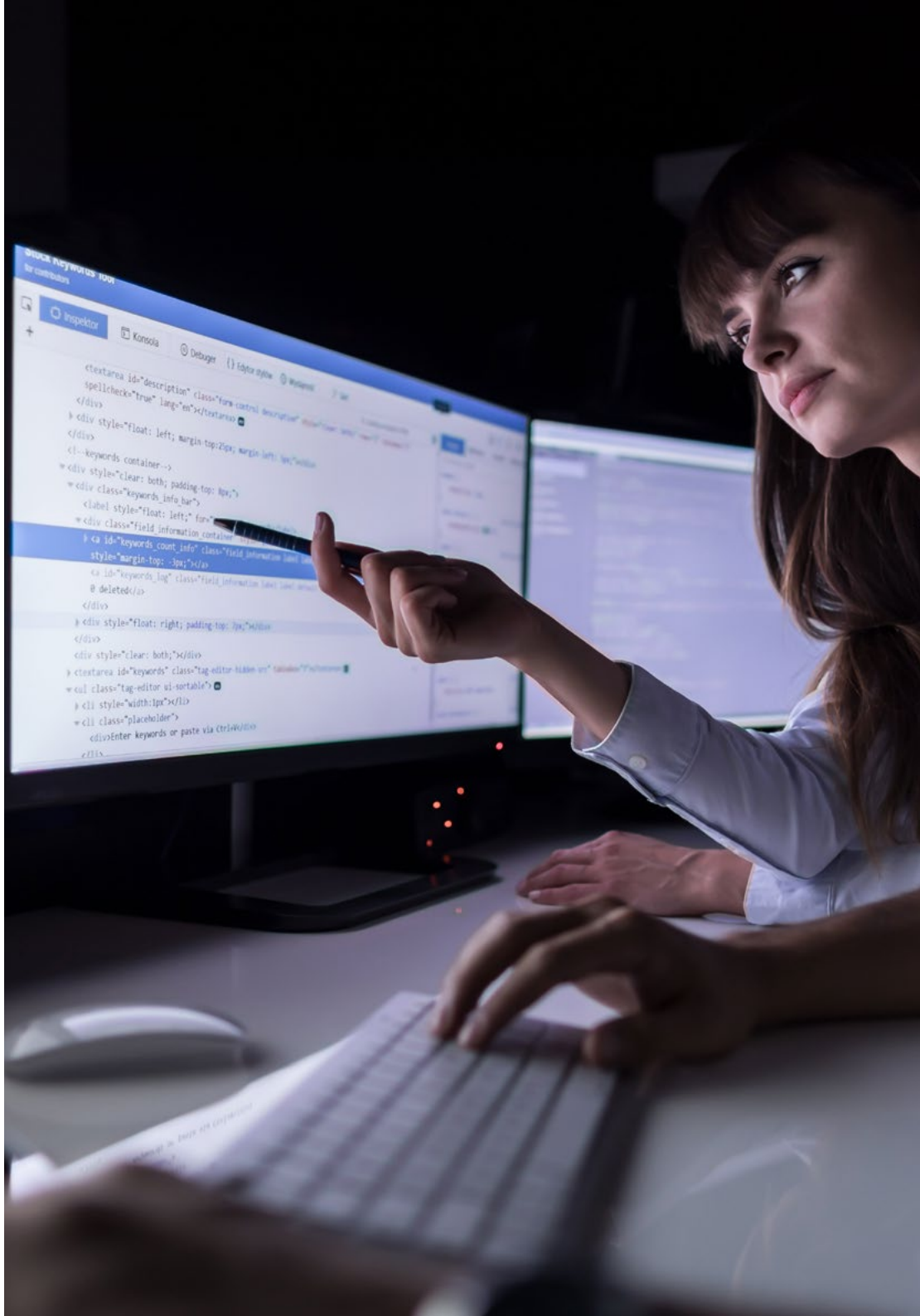
这个校级硕士的主要目标是把学生变成伟大的视频游戏开发者。这个行业正在扩大，越来越需要更多的程序员和更多受过高级培训的专家，所以这个学位是在世界最著名的公司获得巨大职业机会的完美选择。因此，该课程为学生提供了所有必要的技能，使其成为该领域备受追捧的专家，实现其职业生涯的重大和直接提升。

> 有了这个视频游戏编程校级硕士，
> 你的所有梦想现在都触手可及了”

总体目标

- 了解应用于电子游戏的不同编程语言和方法
- 深入了解视频游戏的制作过程和这些阶段的编程整合
- 学习电子游戏设计的基础知识和电子游戏设计师必须了解的理论知识
- 掌握电子游戏中使用的基本编程语言
- 将软件工程和专业编程的知识应用于视频游戏中
- 理解编程在心理病态发展中的作用
- 了解不同的现有游戏机和平台
- 开发网络和多人视频游戏

> 当你完成这个学位时，你将成为你所在环境中最好的视频游戏开发者”



具体目标

模块1.编程基础知识

- 了解计算机的基本结构，软件和通用编程语言
- 分析计算机程序的基本要素，如不同的数据类型，运算符，表达式，语句，输入输出和控制语句
- 解释算法，这是能够开发计算机程序的必要基础

模块2.数据结构和算法

- 学习主要的算法设计策略，以及算法计算的不同方法和措施
- 区分算法的功能，它们的策略和它们在主要已知问题中的使用实例
- 了解Backtracking 技术及其主要用途

模块3.物件导向编程

- 学生将了解面向对象问题的不同设计模式
- 理解软件开发中文档和测试的重要性
- 管理线程和同步化的使用，以及解决并发编程中的常见问题

模块4.游戏机和设备

- 了解主要的输入和输出外围设备的基本操作
- 理解不同平台的主要设计含义
- 研究设备和系统的结构，组织，运作和相互联系
- 了解移动设备和视频游戏平台的操作系统和开发工具包的作用

模块5.软件工程

- 区分软件工程的基础，以及软件过程和包括敏捷技术在内的不同开发模式
- 认识需求工程，它的发展，阐述，谈判和验证，以了解与软件质量和项目管理有关的主要标准

模块6.视频游戏引擎

- 发现电子游戏引擎的功能和架构
- 了解现有游戏引擎的基本特征
- 正确和有效地对应用于视频游戏引擎的应用程序进行编程
- 选择最合适的范式和编程语言，为应用于视频游戏引擎的应用程序

模块7.智能系统

- 建立与代理理论和代理结构及其推理过程有关的概念
- 吸收信息和知识概念背后的理论和实践，以及表现知识的不同方式
- 了解语义推理器，基于知识的系统和专家系统的运作

模块8.实时编程

- 分析实时编程语言区别于传统编程语言的主要特征
- 理解计算机系统的基本概念
- 获得应用主要实时编程基础和技术的能力

模块9.网页游戏的设计和开发

- 学生将能够设计游戏和交互式网络应用程序，并提供相应的文档
- 评估游戏和互动网络应用的主要特点，以便以专业和正确的方式进行交流

模块10.网络和多人游戏系统

- 描述传输控制协议/互联网协议（TCP/IP）架构和无线网络的基本操作
- 分析适用于电子游戏的安全问题
- 获得开发多人在线游戏的能力

03

能力

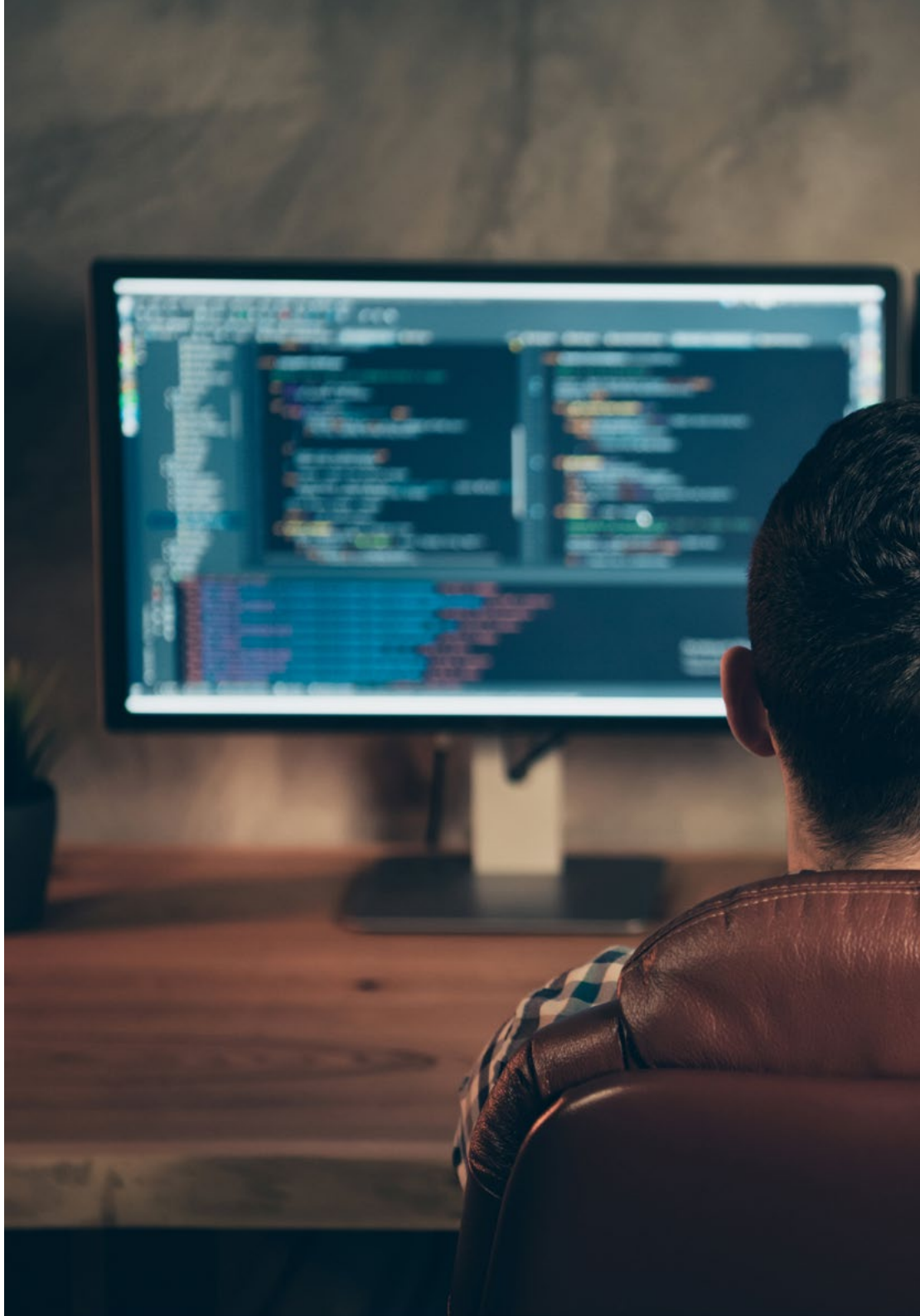
这个视频游戏编程校级硕士使学生成为开发这种视听作品的真正专家，这要归功于它所提供的技能和能力。因此，由于这个优秀的课程，学生将获得一系列专业工具，他们将能够面对与视频游戏编程相关的任何类型的挑战，从而成为其公司的重要人员。

你将掌握视频游戏开发的所有方面”

总体能力

- 设计一个视频游戏的所有阶段，从最初的想法到最后的推出
- 专业从事视频游戏程序员
- 深化所有开发部分，从最初的架构，玩家角色的编程和游戏过程中涉及的所有元素
- 获得项目的整体愿景，能够为视频游戏设计中出现的不同问题和挑战提供解决方案

> 作为一名视频游戏程序员，通过这个校级硕士获得卓越的成就”



具体能力

- 了解成为专业视频游戏开发者的必要软件
- 了解玩家的体验，知道如何分析视频游戏的可玩性
- 了解视频游戏编程过程的所有理论和实践程序
- 掌握对视频游戏世界最有用的编程语言
- 将所学到的编程整合到不同类型的控制台和平台上
- 课程网络和多人视频游戏
- 吸收视频游戏引擎的概念，以便能够正确编程
- 将软件工程和专业编程的知识应用视频游戏中

04

结构和内容

该视频游戏编程校级硕士的内容是由该领域的领先专家团队精心设计的，他们充分了解行业的现状。因此，由于这个课程，学生将能够学习到所有必要的知识，以便能够应对该行业公司的需求，因为他们已经为其复杂性和不断变化的特殊性和特性做了专门准备。

这些内容将使你成为视频游戏编程方面的伟大专家”

模块1.编程基础知识

1.1. 编程简介
- 1.1.1. 计算机的基本结构
- 1.1.2. 软件
- 1.1.3. 编程语言
- 1.1.4. 计算机应用的生命周期

1.2. 算法设计
- 1.2.1. 问题的解决
- 1.2.2. 描述性的技术
- 1.2.3. 算法的要素和结构

1.3. 方案的要素
- 1.3.1. C++语言的起源和特点
- 1.3.2. 开发环境
- 1.3.3. 计划理念
- 1.3.4. 基本数据类型
- 1.3.5. 操作符
- 1.3.6. 表达方式
- 1.3.7. 句子
- 1.3.8. 数据输入和输出

1.4. 控制声明
- 1.4.1. 句子
- 1.4.2. 分叉
- 1.4.3. 循环

1.5. 抽象和模块化。功能
- 1.5.1. 模块化设计
- 1.5.2. 功能和效用的概念
- 1.5.3. 函数的定义
- 1.5.4. 函数调用中的执行流程
- 1.5.5. 功能原型
- 1.5.6. 返回结果
- 1.5.7. 调用一个函数。参数
- 1.5.8. 通过引用和值传递参数
- 1.5.9. 范围标识符

1.6. 数据结构静态
- 1.6.1. Arrays
- 1.6.2. 阵列。多面体
- 1.6.3. 搜索和排序
- 1.6.4. 链字符串的 E/S 函数
- 1.6.5. 结构。連結
- 1.6.6. 新的数据类型

1.7. 数据结构动态性: 指标
- 1.7.1. 概念指标的定义
- 1.7.2. 运算符和指标的操作
- 1.7.3. 指标的数组
- 1.7.4. 指标和数组
- 1.7.5. 指向字符串的指标
- 1.7.6. 指向结构的指标
- 1.7.7. 多方向性
- 1.7.8. 指向函数的指针
- 1.7.9. 将函数，结构和数组作为函数参数传递

1.8. 文件
- 1.8.1. 基本概念
- 1.8.2. 文件操作
- 1.8.3. 文件的类型
- 1.8.4. 文件的组织机构
- 1.8.5. C++文件简介
- 1.8.6. 文件处理

1.9. 递归
- 1.9.1. 递归的定义
- 1.9.2. 递归的类型
- 1.9.3. 优势和劣势
- 1.9.4. 考虑因素
- 1.9.5. 递归-迭代转换
- 1.9.6. 递归栈

1.10. 测试和文件
- 1.10.1. 程序测试
- 1.10.2. 白盒测试
- 1.10.3. 黑匣子测试
- 1.10.4. 测试工具
- 1.10.5. 方案文件

模块2.数据结构和算法

2.1. 算法设计策略介绍
- 2.1.1. 递归
- 2.1.2. 分而治之
- 2.1.3. 其他策略

2.2. 算法的效率和分析
- 2.2.1. 效率措施
- 2.2.2. 测量输入尺寸
- 2.2.3. 测量执行时间
- 2.2.4. 最差，最好和中等情况
- 2.2.5. 渐进式记数法
- 2.2.6. 非递归算法的数学分析标准
- 2.2.7. 递归算法的数学分析
- 2.2.8. 算法的实证分析

2.3. 排序算法
- 2.3.1. 分拣的概念
- 2.3.2. 泡沫分类
- 2.3.3. 通过选择进行排序
- 2.3.4. 按插入法排序
- 2.3.5. 通过合并排序(merge_sort)
- 2.3.6. 快速排序 (quick_sort)

2.4. 树形算法
- 2.4.1. 树木的概念
- 2.4.2. 二进制树
- 2.4.3. 树径
- 2.4.4. 表现表达式
- 2.4.5. 有序二进制树
- 2.4.6. 平衡的二进制树

2.5. 带 Heaps的算法
- 2.5.1. Heaps
- 2.5.2. Heapsort算法
- 2.5.3. 优先级队列

2.6. 图形算法
- 2.6.1. 代表
- 2.6.2. 宽度上的遍历
- 2.6.3. 旅行的深度
- 2.6.4. 拓扑结构的安排

2.7. Greedy的算法
- 2.7.1. Greedy的策略
- 2.7.2. Greedy策略的要素
- 2.7.3. 货币兑换
- 2.7.4. 旅行者问题
- 2.7.5. 背包问题

2.8. 最小化寻路
- 2.8.1. 最小路径问题
- 2.8.2. 负弧和循环
- 2.8.3. 迪克斯特拉的算法

2.9. 图上的Greedy 算法
- 2.9.1. 最小覆盖树
- 2.9.2. 普利姆的算法
- 2.9.3. 克鲁斯卡的算法
- 2.9.4. 复杂度分析

2.10. 溯源
- 2.10.1. Backtracking
- 2.10.2. 替代技术

模块3.物件导向编程

3.1. 物件导向的程序设计简介
- 3.1.1. 物件导向的程序设计简介
- 3.1.2. 班级设计
- 3.1.3. 问题建模的UML介绍

3.2. 类之间的关系
- 3.2.1. 抽象和继承
- 3.2.2. 高级继承概念
- 3.2.3. 多态性
- 3.2.4. 组成和聚集

3.3. 面向对象问题的设计模式介绍
- 3.3.1. 什么是设计模式?
- 3.3.2. 工厂模式
- 3.3.4. 单子模式
- 3.3.5. 观察者模式
- 3.3.6. 复合模式

3.4. 例外情况
- 3.4.1. 什么是例外情况?
- 3.4.2. 异常捕获和处理
- 3.4.3. 抛出异常
- 3.4.4. 异常的产生

3.5. 用户界面
- 3.5.1. Qt简介
- 3.5.2. 定位
- 3.5.3. 什么是活动?
- 3.5.4. 事件:定义和捕捉
- 3.5.5. 用户界面的开发

3.6. 并发编程简介
- 3.6.1. 并发编程简介
- 3.6.2. 过程和线程概念
- 3.6.3. 进程或线程之间的相互作用
- 3.6.4. C++中的线程
- 3.6.5. 并发编程的优势和劣势

3.7. 线程管理和同步化
- 3.7.1. 线程的生命周期
- 3.7.2. Thread类
- 3.7.3. 线程调度
- 3.7.4. 螺纹组
- 3.7.5. 守护进程类型线程
- 3.7.6. 同步
- 3.7.7. 锁定机制
- 3.7.8. 沟通机制
- 3.7.9. 监视器

3.8. 并行编程中的常见问题
- 3.8.1. 消费者-生产者问题
- 3.8.2. 读者和作家的问题
- 3.8.3. 哲学家的晚餐问题

3.9. 软件文档和测试
- 3.9.1. 为什么软件文档很重要?
- 3.9.2. 设计文件
- 3.9.3. 使用文档工具

3.10. 软件测试
- 3.10.1. 软件测试简介
- 3.10.2. 测试的类型
- 3.10.3. 单元测试
- 3.10.4. 集成测试
- 3.10.5. 验证测试
- 3.10.6. 系统测试

模块4.游戏机和设备

4.1. 视频游戏编程的历史
- 4.1.1. 雅达利时期(1977-1985年)
- 4.1.2. NES和SNES时期(1985-1995)
- 4.1.3. PlayStation / PlayStation 2时期(1995-2005)
- 4.1.4. Xbox 360, PS3和Wii时期(2005-2013)
- 4.1.5. Xbox One, PS4和Wii U - Switch时期(2013年至今)
- 4.1.6. 未来

4.2. 电子游戏中的游戏性的历史
- 4.2.1. 介绍
- 4.2.2. 社会背景
- 4.2.3. 结构图
- 4.2.4. 未来

4.3. 适应现代社会的发展
- 4.3.1. 基于运动的游戏
- 4.3.2. 虚拟现实
- 4.3.3. 扩增实境
- 4.3.4. 混合现实

4.4. 统一性 脚本 I 和示例
- 4.4.1. 什么是脚本?
- 4.4.2. 我们的第一个脚本
- 4.4.3. 添加脚本
- 4.4.4. 打开脚本
- 4.4.5. 猴子行为
- 4.4.6. 调试

4.5. 统一性Scripting II和实例
- 4.5.1. 键盘和鼠标输入
- 4.5.2. 光线投射
- 4.5.3. 实例化
- 4.5.4. 可变因素
- 4.5.5. 公共变量和序列化变量

4.6. 统一性Scripting III和实例
- 4.6.1. 获取组件
- 4.6.2. 修改组件
- 4.6.3. 测试
- 4.6.4. 多个对象
- 4.6.5. 对撞机和触发器
- 4.6.6. 四元数

4.7. 外围设备
- 4.7.1. 演变和分类
- 4.7.2. 外围设备和接口
- 4.7.3. 当前外设
- 4.7.4. 下一个未来

4.8. 电子游戏:未来的前景
- 4.8.1. 基于云的游戏
- 4.8.2. 无人驾驶
- 4.8.3. 沉浸式现实
- 4.8.4. 其他替代品

4.9. 建筑学
- 4.9.1. 电子游戏的特殊需求
- 4.9.2. 架构的演变
- 4.9.3. 当前架构
- 4.9.4. 架构之间的差异

4.10. 开发套件及其演变
- 4.10.1. 介绍
- 4.10.2. 第三代开发套件
- 4.10.3. 第四代开发套件
- 4.10.4. 第五代开发套件
- 4.10.5. 第六代开发套件

模块5.软件工程

5.1. 软件工程与建模导论
- 5.1.1. 软件的本质
- 5.1.2. Webapps 的独特性
- 5.1.3. 软件工程
- 5.1.4. 软件过程
- 5.1.5. 软件工程实践
- 5.1.6. 软件神话
- 5.1.7. 这一切是如何开始的
- 5.1.8. 物件导向概念
- 5.1.9. UML 简介

5.2. 软件过程
- 5.2.1. 通用过程模型
- 5.2.2. 规范的过程模型
- 5.2.3. 专门的过程模型
- 5.2.4. 统一过程
- 5.2.5. 个人和团队过程模型
- 5.2.6. 什么是敏捷?
- 5.2.7. 什么是敏捷过程?
- 5.2.8. Scrum
- 5.2.9. 敏捷过程工具包

5.3. 指导软件工程实践的原则
- 5.3.1. 指导过程的原则
- 5.3.2. 指导实践的原则
- 5.3.3. 沟通原则
- 5.3.4. 规划原则
- 5.3.5. 建模原则
- 5.3.6. 施工原则
- 5.3.7. 部署原则

5.4. 了解需求
- 5.4.1. 需求工程
- 5.4.2. 建立基础
- 5.4.3. 需求调查
- 5.4.4. 用例开发
- 5.4.5. 需求模型的细化
- 5.4.6. 需求协商
- 5.4.7. 需求验证

5.5. 需求建模：场景，信息和分析类
- 5.5.1. 需求分析
- 5.5.2. 基于场景的建模
- 5.5.3. 提供用例的 UML 模型
- 5.5.4. 数据建模概念
- 5.5.5. 基于类的建模
- 5.5.6. 类图

5.6. 需求建模：流程，行为和模式
- 5.6.1. 为策略建模的要求
- 5.6.2. 面向流的建模
- 5.6.3. 状态图
- 5.6.4. 创建行为模型
- 5.6.5. 序列图
- 5.6.6. 通讯图
- 5.6.7. 需求建模模式

5.7. 设计理念
- 5.7.1. 软件工程背景下的设计
- 5.7.2. 设计过程
- 5.7.3. 设计理念
- 5.7.4. 面向对象的设计理念
- 5.7.5. 设计模型

5.8. 架构设计
- 5.8.1. 软件架构
- 5.8.2. 架构类型
- 5.8.3. 架构风格
- 5.8.4. 架构设计
- 5.8.5. 架构替代设计的演变
- 5.8.6. 使用数据流映射架构

5.9. 在组件级别和基于模式的设计
- 5.9.1. 什么是组件？
- 5.9.2. 照明技术基于类的组件设计
- 5.9.3. 在组件级别执行设计
- 5.9.4. 传统组件设计
- 5.9.5. 基于组件的开发
- 5.9.6. 设计模式
- 5.9.7. 基于模式的软件设计
- 5.9.8. 架构模式
- 5.9.9. 组件级设计模式
- 5.9.10. 用户界面设计模式

5.10. 软件质量和项目管理
- 5.10.1. 质量
- 5.10.2. 软件质量
- 5.10.3. 软件质量困境
- 5.10.4. 实现软件质量
- 5.10.5. 软件质量保证
- 5.10.6. 行政范围
- 5.10.7. 职员
- 5.10.8. 产品
- 5.10.9. 过程
- 5.10.10. 项目
- 5.10.11. 原则与实践

模块6.视频游戏引擎

6.1. 视频游戏和信息通信技术
- 6.1.1. 介绍
- 6.1.2. 机会
- 6.1.3. 挑战
- 6.1.4. 结论

6.2. 视频游戏引擎的历史
- 6.2.1. 介绍
- 6.2.2. 雅达利时代
- 6.2.3. 80年代时代
- 6.2.4. 第一引擎。90年代时代
- 6.2.5. 当前引擎

6.3. 视频游戏引擎
- 6.3.1. 引擎的类型
- 6.3.2. 视频游戏引擎的部件
- 6.3.3. 当前引擎
- 6.3.4. 为我们的项目选择一个引擎

6.4. 游戏制作引擎
- 6.4.1. 介绍
- 6.4.2. 情景设计
- 6.4.3. 精灵图和动画
- 6.4.4. 碰撞
- 6.4.5. GML中的脚本

6.5. 虚幻引擎4。简介
- 6.5.1. 什么是虚幻引擎4?它的理念是什么?
- 6.5.2. 材料
- 6.5.3. 介面
- 6.5.4. 动画片
- 6.5.5. 粒子系统
- 6.5.6. 人工智能
- 6.5.7. FPS

6.6. 虚幻引擎4。可视化脚本
- 6.6.1. Blueprints和可视化Scripting的哲学
- 6.6.2. 调试
- 6.6.3. 变量的类型
- 6.6.4. 基本流量控制

6.7. Unity 5引擎
- 6.7.1. 用C#和Visual Studio编程
- 6.7.2. 创建预制板
- 6.7.3. 使用 Gizmos来控制电子游戏
- 6.7.4. 自适应的引擎2D 和 3D

6.8. 戈多引擎
- 6.8.1. 戈多的设计理念
- 6.8.2. 物件导向的设计和组合
- 6.8.3. 全部包含在一个包中
- 6.8.4. 免费和社区驱动的软件

6.9. RPG Maker引擎
- 6.9.1. RPG Maker哲学
- 6.9.2. 以此作为参考
- 6.9.3. 创造一个有个性的游戏
- 6.9.4. 成功的商业游戏

6.10. 来源2引擎
- 6.10.1. 来源2 哲学
- 6.10.2. 来源和来源2：发展
- 6.10.3. 社区使用视听内容和视频游戏
- 6.10.4. 来源2引擎的未来
- 6.10.5. Mods和成功的游戏

模块7.智能系统

7.1. 代理理论
- 7.1.1. 概念历史
- 7.1.2. 代理定义
- 7.1.3. 人工智能中的代理
- 7.1.4. 软件工程中的代理

7.2. 代理架构
- 7.2.1. 代理的推理过程
- 7.2.2. 反应剂
- 7.2.3. 演绎代理
- 7.2.4. 混合代理
- 7.2.5. 比较

7.3. 信息和知识
- 7.3.1. 数据，信息和知识之间的区别
- 7.3.2. 数据质量评估
- 7.3.3. 数据采集方法
- 7.3.4. 信息获取方法
- 7.3.5. 知识获取方法

7.4. 知识表示
- 7.4.1. 知识所代表的重要性
- 7.4.2. 通过其角色定义知识表示
- 7.4.3. 知识表示的特征

7.5. 本体论
- 7.5.1. 元数据简介
- 7.5.2. 本体论的哲学概念
- 7.5.3. 本体计算概念
- 7.5.4. 领域本体和顶级本体
- 7.5.5. 如何构建本体

7.6. 本体语言和本体创建软件
- 7.6.1. RDF，Turtle 和 N3 三元组
- 7.6.2. RDF模式
- 7.6.3. OWL
- 7.6.4. SPARQL
- 7.6.5. 介绍用于创建本体的不同工具
- 7.6.6. Protégé安装和使用

7.7. 语义网
- 7.7.1. 语义网的当前和未来状态
- 7.7.2. 语义网应用

7.8. 其他知识表示模型
- 7.8.1. 词汇
- 7.8.2. 全球视野
- 7.8.3. 分类法
- 7.8.4. 叙词表
- 7.8.5. 大众分类法
- 7.8.6. 比较
- 7.8.7. 心理地图

7.9. 知识表示的评估和整合
- 7.9.1. 零阶逻辑
- 7.9.2. 一阶逻辑
- 7.9.3. 描述性逻辑
- 7.9.4. 不同类型逻辑之间的关系
- 7.9.5. 序言：基于一阶逻辑的编程

7.10. 语义推理器，基于知识的系统和专家系统
- 7.10.1. 推理者的概念
- 7.10.2. 推理机的应用
- 7.10.3. 基于知识的系
- 7.10.4. MYCIN，专家系统的历史
- 7.10.5. 专家系统的元素和架构
- 7.10.6. 专家系统的创建

模块8.实时编程

8.1. 并发编程基础
- 8.1.1. 基本概念
- 8.1.2. 并发
- 8.1.3. 并发的好处
- 8.1.4. 并发和硬件

8.2. Java 中的基本并发支持结构
- 8.2.1. Java 中的并发
- 8.2.2. 线程创建
- 8.2.3. 方法
- 8.2.4. 同步

8.3. Threads, 生命周期, 优先级, 中断, 状态, 执行程序
- 8.3.1. 线程
- 8.3.2. 生命周期
- 8.3.3. 优先事项
- 8.3.4. 中断
- 8.3.5. 状况
- 8.3.6. 执行者

8.4. 互斥
- 8.4.1. 什么是互斥?
- 8.4.2. 德克算法
- 8.4.3. 彼得森算法
- 8.4.4. Java中的互斥

8.5. 状态依赖
- 8.5.1. 依赖注入
- 8.5.2. Java中模式的实现
- 8.5.3. 注入依赖的方法
- 8.5.4. 例子

8.6. 设计模式
- 8.6.1. 介绍
- 8.6.2. 创作模式
- 8.6.3. 结构模式
- 8.6.4. 行为模式

8.7. 使用 Java 库
- 8.7.1. Java 中的库是什么?
- 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. 着色器编程
- 8.8.1. 管道 3D 和栅格
- 8.8.2. 顶点着色
- 8.8.3. Pixel Shading: 照明I
- 8.8.4. Pixel Shading: 照明II
- 8.8.5. Post-Effectos

8.9. 实时编程
- 8.9.1. 介绍
- 8.9.2. 中断处理
- 8.9.3. 进程之间的同步和通信
- 8.9.4. 实时规划系统

8.10. 实时规划
- 8.10.1. 概念
- 8.10.2. 实时系统参考模型
- 8.10.3. 规划政策
- 8.10.4. 循环规划器
- 8.10.5. 具有静态属性的调度程序
- 8.10.6. 具有动态属性的调度程序

模块9.网页游戏的设计和开发

9.1. 网络起源和标准
- 9.1.1. 互联网起源
- 9.1.2. 万维网的创建
- 9.1.3. 网络标准的出现
- 9.1.4. 网络标准的兴起

9.2. HTTP 和客户端-服务器结构
- 9.2.1. 客户端-服务器角色
- 9.2.2. 客户端-服务器通信
- 9.2.3. 最近的历史
- 9.2.4. 集中计算

9.3. 网络编程:介绍
- 9.3.1. 基本概念
- 9.3.2. 准备 Web 服务器
- 9.3.3. HTML5 基础知识
- 9.3.4. HTML 表单

9.4. HTML 简介和示例
- 9.4.1. HTML5 历史
- 9.4.2. HTML5 元素
- 9.4.3. APIS
- 9.4.4. CCS3

9.5. 文档对象模型
- 9.5.1. 什么是文档对象模型?
- 9.5.2. 使用文档类型
- 9.5.3. 验证 HTML 的重要性
- 9.5.4. 存取元素
- 9.5.5. 创建元素和文本
- 9.5.6. 使用内部 HTML
- 9.5.7. 删除元素或文本节点
- 9.5.8. 读取和写入元素的属性
- 9.5.9. 操作元素样式
- 9.5.10. 一次附加多个文件

9.6. CSS 简介和示例
- 9.6.1. CSS3 语法
- 9.6.2. 样式表
- 9.6.3. 标签
- 9.6.4. 选择器
- 9.6.5. 使用 CSS 进行网页设计

9.7. Javascript 简介和示例
- 9.7.1. 什么是 JavaScript?
- 9.7.2. 语言简史
- 9.7.3. Javascript 版本
- 9.7.4. 显示对话框
- 9.7.5. JavaScript 语法
- 9.7.6. Script理解
- 9.7.7. 空间
- 9.7.8. 注释
- 9.7.9. 功能
- 9.7.10. 页面和外部的 Javascript

9.8. Javascript 中的函数
- 9.8.1. 函数声明
- 9.8.2. 函数表达式
- 9.8.3. 呼叫功能
- 9.8.4. 递归
- 9.8.5. 嵌套函数和闭包
- 9.8.6. 变量保存
- 9.8.7. 多嵌套函数
- 9.8.8. 命名冲突
- 9.8.9. 闭包或闭包
- 9.8.10. 函数的参数

9.9. 用于开发 Web 游戏的 PlayCanvas
- 9.9.1. 什么是 PlayCanvas?
- 9.9.2. 创建对象
- 9.9.3. 添加物理
- 9.9.4. 添加模型
- 9.9.5. 更改重力和场景设置
- 9.9.6. 运行 Scripts
- 9.9.7. 相机控制

9.10. Phaser 开发网页游戏
- 9.10.1. 什么是移相器?
- 9.10.2. 加载资源
- 9.10.3. 建设世界
- 9.10.4. 平台
- 9.10.5. 玩家
- 9.10.6. 添加物理
- 9.10.7. 使用键盘
- 9.10.8. 拾起 Pickups
- 9.10.9. 积分和分数
- 9.10.10. 弹跳炸弹

模块10.网络和多人游戏系统

10.1. 多人视频游戏的历史和演变
- 10.1.1. 1970 年代:第一款多人游戏
- 10.1.2. 90年代:Duke Nukem, Doom, Quake
- 10.1.3. 多人视频游戏热潮
- 10.1.4. 本地和在线多人游戏
- 10.1.5. 派对游戏

10.2. 多人商业模式
- 10.2.1. 新兴商业模式的起源与运作
- 10.2.2. 在线销售服务
- 10.2.3. 免费玩
- 10.2.4. 小额支付
- 10.2.5. 宣传
- 10.2.6. 每月付款订阅
- 10.2.7. 每场比赛付费
- 10.2.8. 先试后买

10.3. Jocal 游戏和在线游戏
- 10.3.1. 本地游戏:开始
- 10.3.2. 派对游戏:任天堂与家族的联合
- 10.3.3. 网络游戏:开始
- 10.3.4. 网络游戏的演变

10.4. OSI 型号:第一层
- 10.4.1. OSI模型:简介
- 10.4.2. 物理层
- 10.4.3. 数据链路层
- 10.4.4. 网络层

10.5. OSI 型号：第二层
- 10.5.1. 传输层
- 10.5.2. 会话层
- 10.5.3. 表示层
- 10.5.4. 应用层

10.6. 计算机网络和互联网
- 10.6.1. 什么是计算机网络?
- 10.6.2. 软件
- 10.6.3. 硬件
- 10.6.4. 服务器
- 10.6.5. 网络存储
- 10.6.6. 网络协议

10.7. 移动和无线网络
- 10.7.1. 移动网络
- 10.7.2. 无线网络
- 10.7.3. 移动网络运营
- 10.7.4. 数字技术

10.8. 安全
- 10.8.1. 个人安全
- 10.8.2. 电子游戏中的黑客和作弊
- 10.8.3. 反作弊安全
- 10.8.4. 反作弊安全系统分析

10.9. 多人系统:服务器
- 10.9.1. 服务器托管
- 10.9.2. MMO视频游戏
- 10.9.3. 专用视频游戏服务器
- 10.9.4. 局域网

10.10. 多人视频游戏设计和编程
- 10.10.1. 虚幻中多人视频游戏设计的基础知识
- 10.10.2. Unity 中多人视频游戏设计的基础知识
- 10.10.3. 如何让多人游戏变得有趣?
- 10.10.4. 超越命令:多人游戏控制的创新

> 如果你想发展一个伟大的职业编程世界各地著名的电子游戏,这就是你正在寻找的学位”

05

方法

这个培训计划提供了一种不同的学习方式。我们的方法是通过循环的学习模式发展起来的:**再学习**。

这个教学系统被世界上一些最著名的医学院所采用，并被**新英格兰医学杂志**等权威出版物认为是最有效的教学系统之一。

发现再学习，这个系统放弃了传统的线性学习，带你体验循环教学系统：这种学习方式已经证明了其巨大的有效性，尤其是在需要记忆的科目中”

案例研究,了解所有内容的背景

我们的方案提供了一种革命性的技能和知识发展方法。我们的目标是在一个不断变化,竞争激烈和高要求的环境中加强能力建设。

> 和TECH,你可以体验到一种正在动摇世界各地传统大学基础的学习方式”

你将进入一个以重复为基础的学习系统,在整个教学大纲中采用自然和渐进式教学。

学生将通过合作活动和真实案例，学习如何解决真实商业环境中的复杂情况。

一种创新并不同的学习方法

该技术课程是一个密集的教学计划，从零开始，提出了该领域在国内和国际上最苛刻的挑战和决定。由于这种方法，个人和职业成长得到了促进，向成功迈出了决定性的一步。案例法是构成这一内容的技术基础，确保遵循当前经济，社会和职业现实。

> 我们的课程使你准备好在不确定的环境中面对新的挑战，并取得事业上的成功”

在世界顶级计算机科学学校存在的时间里，案例法一直是最广泛使用的学习系统。1912年开发的案例法是为了让法律学生不仅在理论内容的基础上学习法律，案例法向他们展示真实的复杂情况，让他们就如何解决这些问题作出明智的决定和价值判断。1924年，它被确立为哈佛大学的一种标准教学方法。

在特定情况下，专业人士应该怎么做?这就是我们在案例法中面对的问题，这是一种以行动为导向的学习方法。在整个课程中，学生将面对多个真实的案例。他们必须整合所有的知识，研究，论证和捍卫他们的想法和决定。

再学习方法

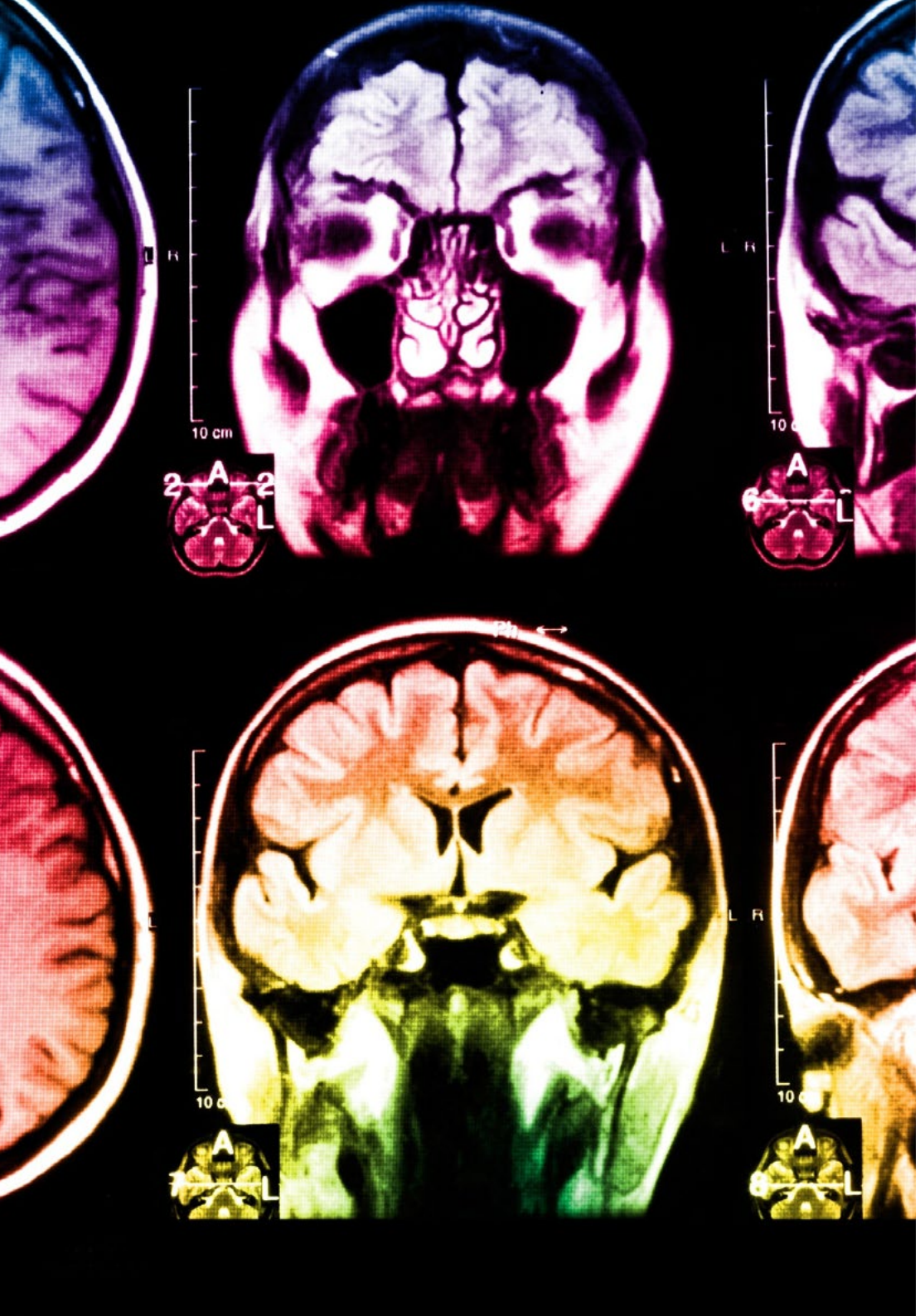
TECH有效地将案例研究方法与基于循环的100%在线学习系统相结合，在每节课中结合了个不同的教学元素。

我们用最好的100%在线教学方法加强案例研究：再学习。

> 在2019年，我们取得了世界上所有西班牙语在线大学中最好的学习成绩。

在TECH，你将用一种旨在培训未来管理人员的尖端方法进行学习。这种处于世界教育学前沿的方法被称为再学习。

我校是唯一获准使用这一成功方法的西班牙语大学。2019年，我们成功地提高了学生的整体满意度（教学质量，材料质量，课程结构，目标……），与西班牙语最佳在线大学的指标相匹配。



在我们的方案中，学习不是一个线性的过程，而是以螺旋式的方式发生（学习，解除学习，忘记和重新学习）。因此，我们将这些元素中的每一个都结合起来。这种方法已经培养了超过65万名大学毕业生，在生物化学，遗传学，外科，国际法，管理技能，体育科学，哲学，法律，工程，新闻，历史，金融市场和工具等不同领域取得了前所未有的成功。所有这些都是在一个高要求的环境中进行的，大学学生的社会经济状况很好，平均年龄为43.5岁。

再学习将使你的学习事半功倍，表现更出色，使你更多地参与到训练中，培养批判精神，捍卫论点和对比意见：直接等同于成功。

从神经科学领域的最新科学证据来看，我们不仅知道如何组织信息，想法，图像y记忆，而且知道我们学到东西的地方和背景，这是我们记住它并将其储存在海马体的根本原因，并能将其保留在长期记忆中。

通过这种方式，在所谓的神经认知背景依赖的电子学习中，我们课程的不同元素与学员发展其专业实践的背景相联系。

该方案提供了最好的教育材料，为专业人士做了充分准备：

学习材料

所有的教学内容都是由教授该课程的专家专门为该课程创作的，因此，教学的发展是具体的。

然后，这些内容被应用于视听格式，创造了TECH在线工作方法。所有这些，都是用最新的技术，提供最高质量的材料，供学生使用。

大师课程

有科学证据表明第三方专家观察的有用性。

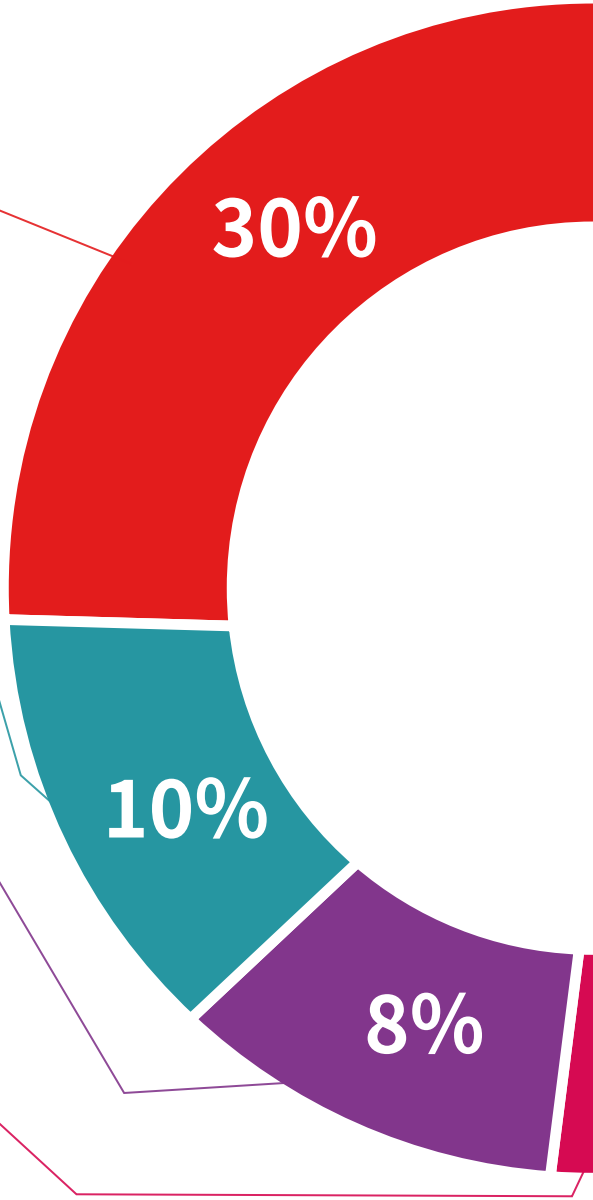
向专家学习可以加强知识和记忆，并为未来的困难决策建立信心。

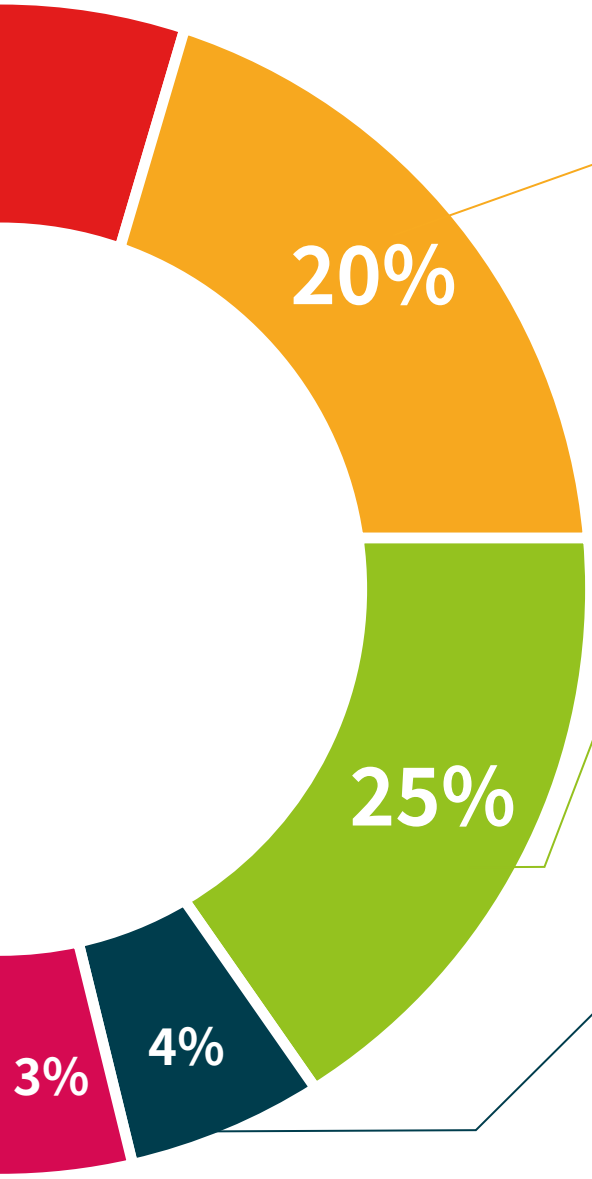
技能和能力的实践

你将开展活动以发展每个学科领域的具体能力和技能。在我们所处的全球化框架内，我们提供实践和氛围帮你取得成为专家所需的技能和能力。

延伸阅读

最近的文章，共识文件和国际准则等。在TECH的虚拟图书馆里，学生可以获得他们完成培训所需的一切。





案例研究

他们将完成专门为这个学位选择的最佳案例研究。由国际上最好的专家介绍,分析和辅导案例。

互动式总结

TECH团队以有吸引力和动态的方式将内容呈现在多媒体丸中,其中包括音频,视频,图像,图表和概念图,以强化知识。

这个用于展示多媒体内容的独特教育系统被微软授予“欧洲成功案例”称号。

测试和循环测试

在整个课程中,通过评估和自我评估活动和练习,定期评估和重新评估学习者的知识:通过这种方式,学习者可以看到他/她是如何实现其目标的。

06

学位

视频游戏编程校级硕士课程除了保证最严格和最新的培训外，还可以获得由TECH科技大学颁发的校级硕士学位证书。

成功地完成这一项目，并获得你的大学学位，没有旅行或行政文书的麻烦”

这个**视频游戏编程校级硕士**包含了市场上最完整和最新的课程。

评估通过后，学生将通过邮寄收到**TECH科技大学**颁发的相应的**校级硕士**学位。

学位由**TECH科技大学**颁发，证明在校级硕士学位中所获得的资质，并满足工作交流，竞争性考试和职业评估委员会的要求。

学位：**视频游戏编程校级硕士**

官方学时：**1,500小时**


tech 科学技术大学

授予

文凭

_____先生/女士，身份证号码_____

顺利通过并获得认证的课程是

校级硕士

视频游戏编程

这是一个由本大学授予的学位，相当于1,500个小时，
开始日期是 dd/mm/aaaa,结束日期是dd/mm/aaaa。

截至2018年6月28日，TECH是一所被公共教育部认可的私立高等教育机构。

2020年六月17日

Tere Guevara Navarro女士
校长

Unique TECH Code: AFWORD23S techtitute.com/certificates


视频游戏编程校级硕士

科目类型	小时
必修(OB)	1,500
选修 (OP)	0
外部实习 (PR)	0
硕士论文 (TFM)	0
总计	1,500

教学大纲概况

学期	学科	小时	性质
1°	编程基础知识	150	必修
1°	数据结构和算法	150	必修
1°	物件导向编程	150	必修
1°	游戏机和设备	150	必修
1°	模块5.软件工程	150	必修
1°	视频游戏引擎	150	必修
1°	智能系统	150	必修
1°	实时编程	150	必修
1°	网页游戏的设计和开发	150	必修
1°	网络和多人游戏系统	150	必修

Tere Guevara Navarro女士
校长

tech 科学技术大学

*海牙认证。如果学生要求对其纸质证书进行海牙认证，TECH EDUCATION将作出必要的安排，并收取认证费用。

tech 科学技术大学

校级硕士
视频游戏编程

» 模式:在线
» 时间:12个月
» 学历:TECH科技大学
» 时间:16小时/周
» 时间表:按你方便的
» 考试:在线

校级硕士
视频游戏编程

tech 科学技术大学