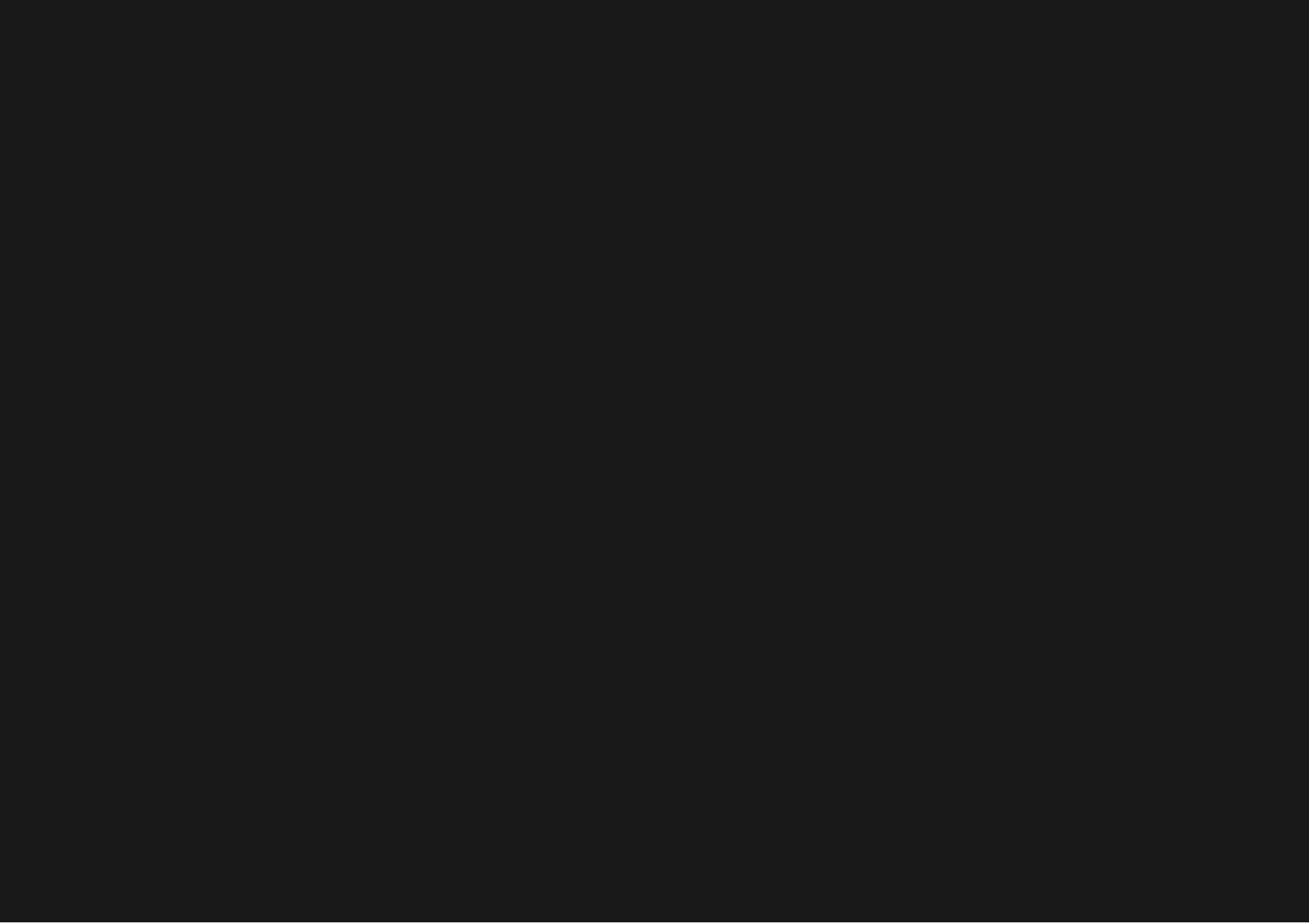


# YEARLY PROJECT

By @raptazure - Haoran Liu

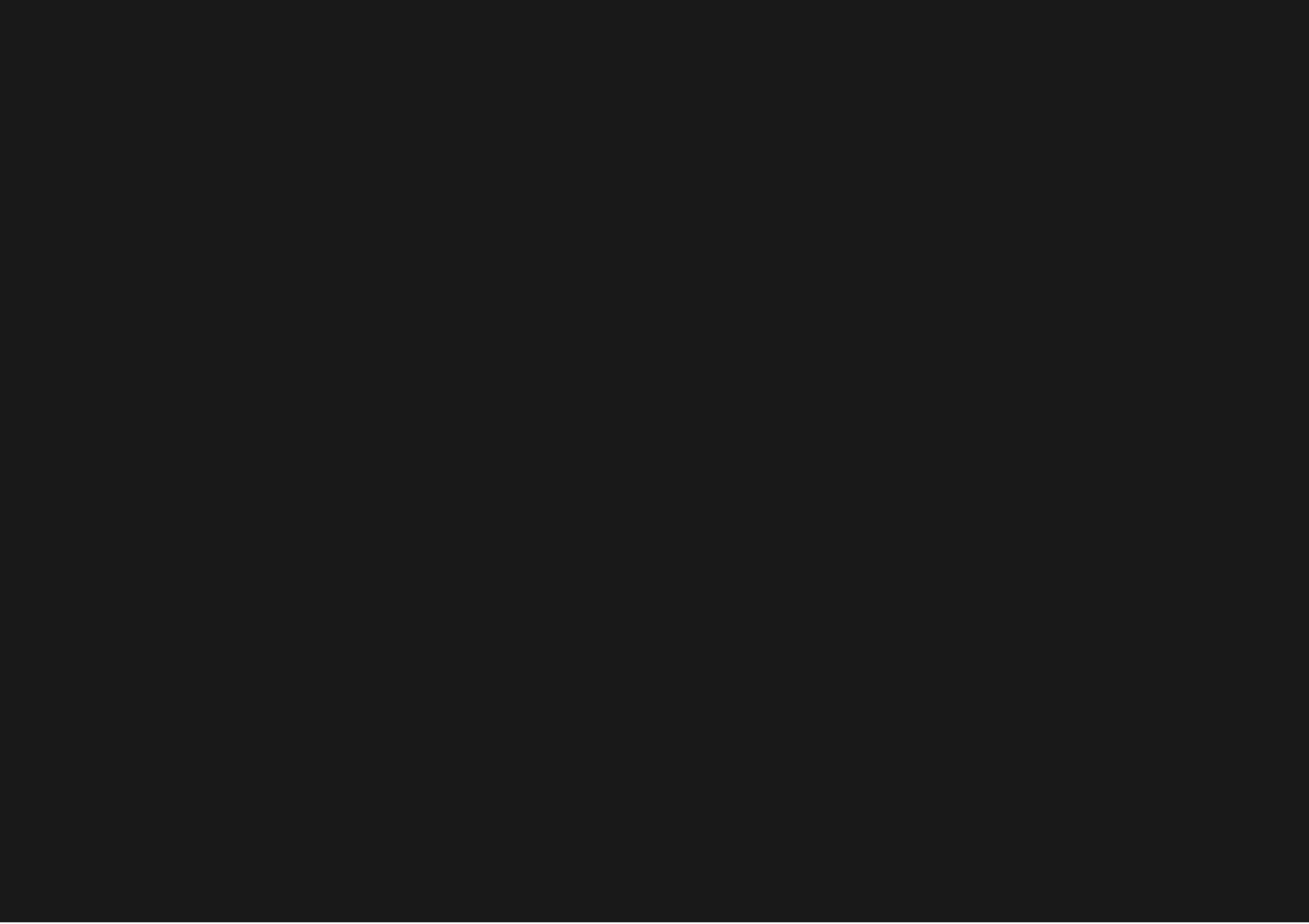
Source Code of the Slide





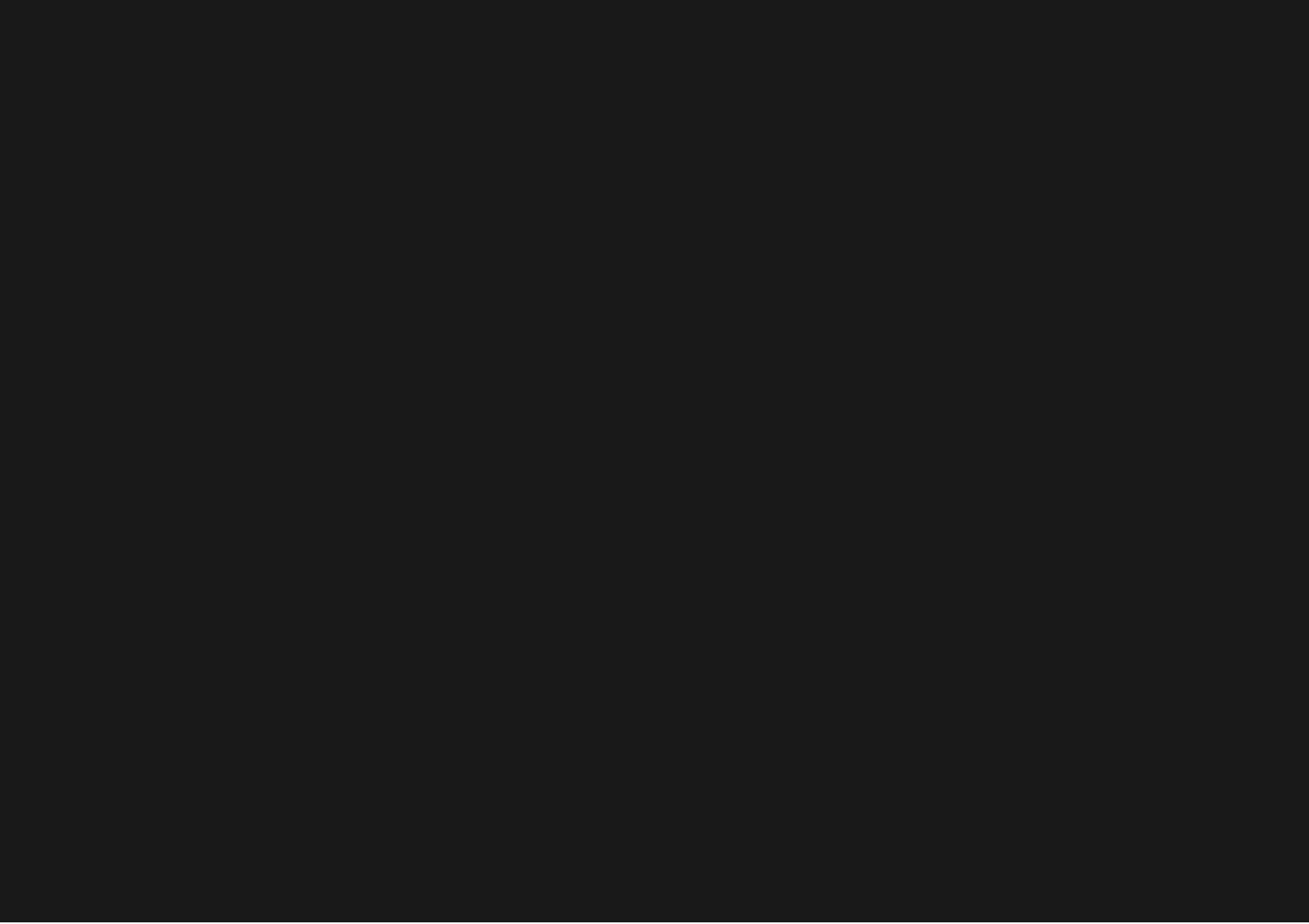
# I. CHOOSE YOUR FIELDS

- Machine Learning
- Embedded Systems
- Web Applications
- Computer Graphics
- Operating Systems
- Distributed Systems
- Databases and Storage System
- Programming Language Theory
- ...



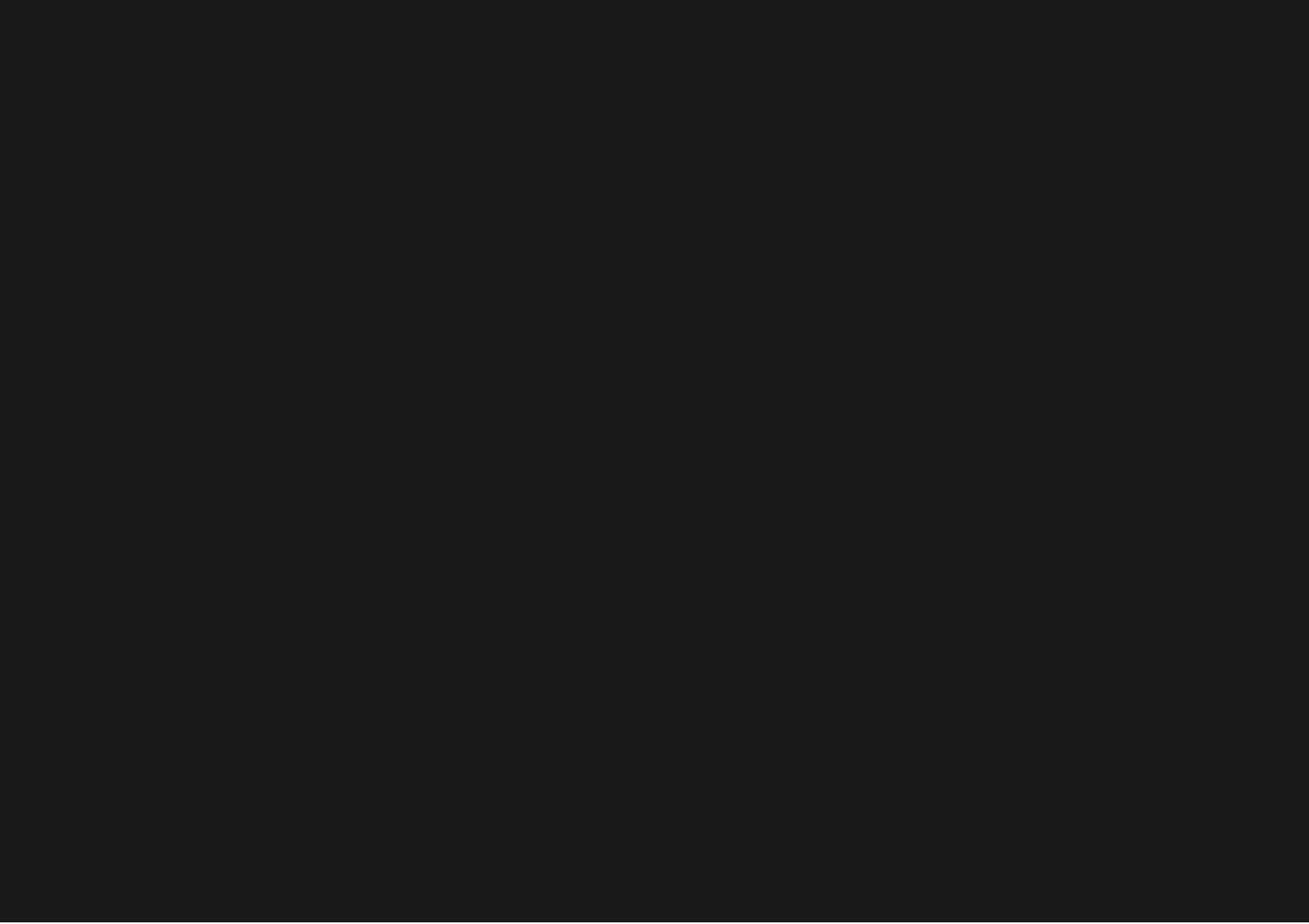
# MACHINE LEARNING

- Video: [Andrew Ng Machine Learning](#)
- Frameworks: [tensorflow](#), [pytorch](#)
- Computer vision: OpenCV(cv2), YOLOv3
- Not only Python (Maybe JS or Rust?)



# EMBEDDED SYSTEMS

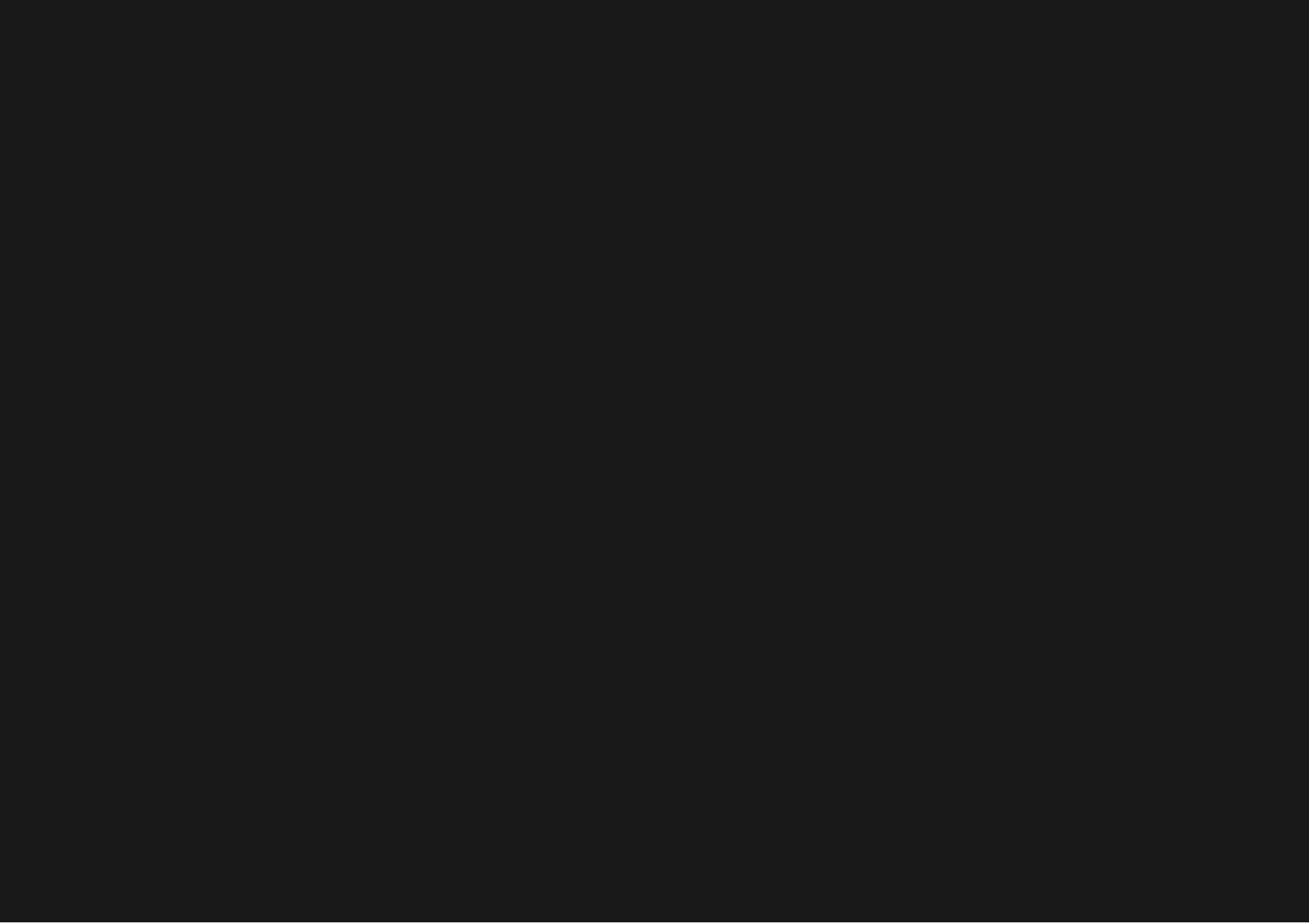
- Raspberry Pi: <https://www.raspberrypi.org/>
- Linux and Open Source
- The Rust Programming Language
  - Rust is blazingly fast and memory-efficient: with no runtime or garbage collector, it can power performance-critical services, run on embedded devices, and easily integrate with other languages.
  - Rust's rich type system and ownership model guarantee memory-safety and thread-safety – enabling you to eliminate many classes of bugs at compile-time.
  - Rust has great documentation, a friendly compiler with useful error messages, and top-notch tooling – an integrated package manager and build tool, smart multi-editor support with auto-completion and type inspections, an auto-formatter, and more.
  - [The Embedded Rust Book](#) | [Chinese version](#)





# WEB APPLICATIONS

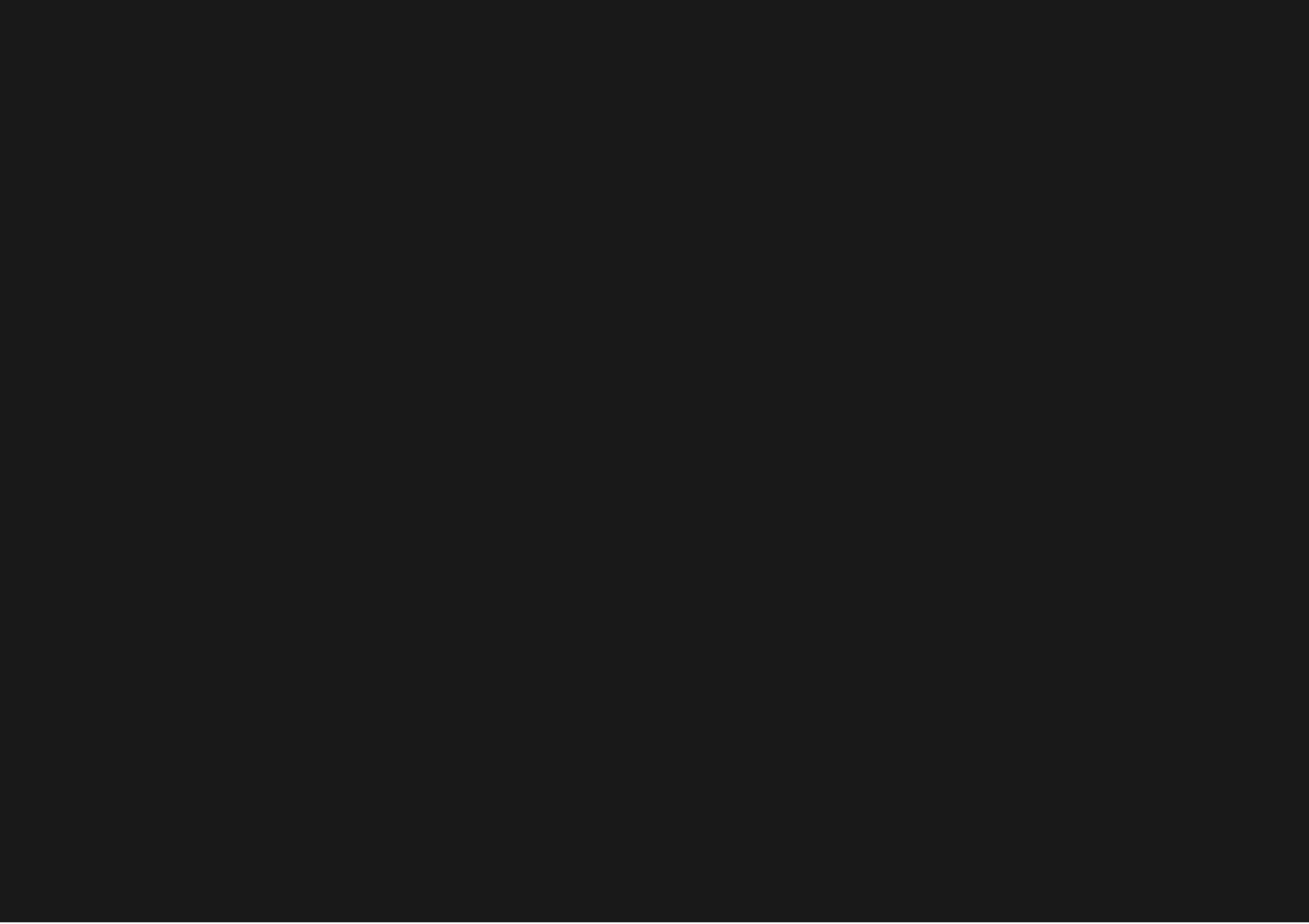
- Goals are important: what do people need?
- Front-end:
  - HTML, CSS, JavaScript → TypeScript
  - Framework/Library: Vue, React, Angular
  - Reactive, Functional: rxjs, elm, OCaml...
- Back-end:
  - RESTful API design, MVC
  - Frameworks: Node.js(express, nest), Java(Spring), Python(flask, django), Ruby(rails), elixir(phoenix)...
  - Databases: MySQL, PostgreSQL, MongoDB...
  - Container: Docker, Kubernetes...
- Mobile Apps:
  - Native: Java(Kotlin) for Android, Objective-C(Swift) for IOS
  - Hybrid: React Native, Flutter (go back to front-end)
- Desktop Apps: Cross-platform solution - QT, Electron...



# COMPUTER GRAPHICS

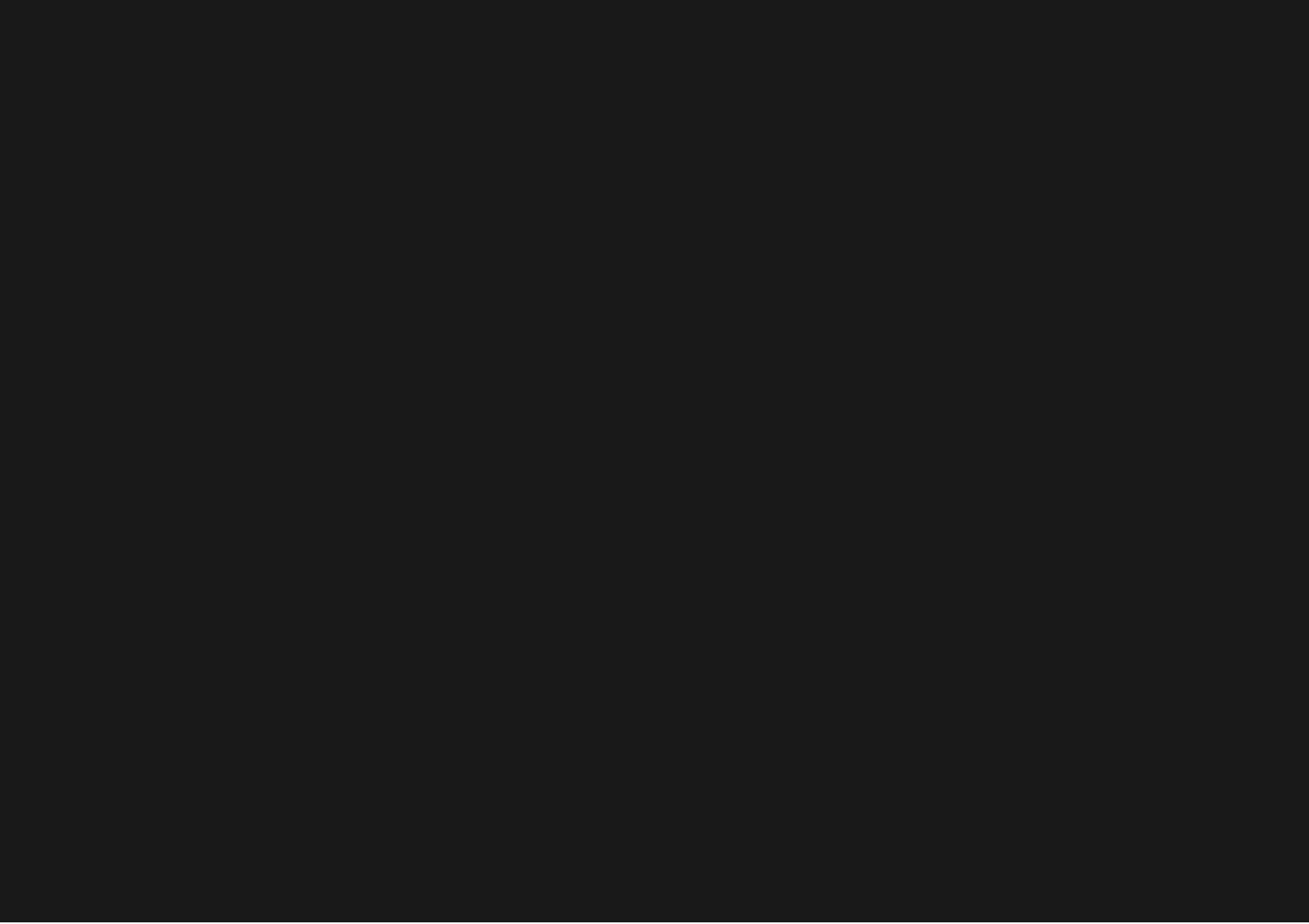
- Rust implementation of “Ray Tracing in One Weekend”
- Games: Unity, UE4...





# OPERATING SYSTEMS

- `Tutorial for rCore OS step by step (2nd edition)`
- `Writing an OS in Rust - Philipp Oppermann's blog`



# DISTRIBUTED STORAGE SYSTEM

- MIT 6.824: Distributed Systems
- PingCAP Talent Plan: TiDB and TiKV

# 如何参与 Talent Plan 课程的学习?

## Step 1

结合个人兴趣爱好及知识背景，选择适合自己的学习路径



### 路径（一）

实现一个 Mini 版本的分布式关系型数据库



### 路径（二）

实现一个 Mini 版本的分布式 Key-value 数据库



### 路径（三）

参与工业级开源分布式关系型数据库 TiDB 的开发实践



### 路径（四）

参与工业级开源分布式 Key-value 数据库 TiKV 的开发实践



### 路径（五）

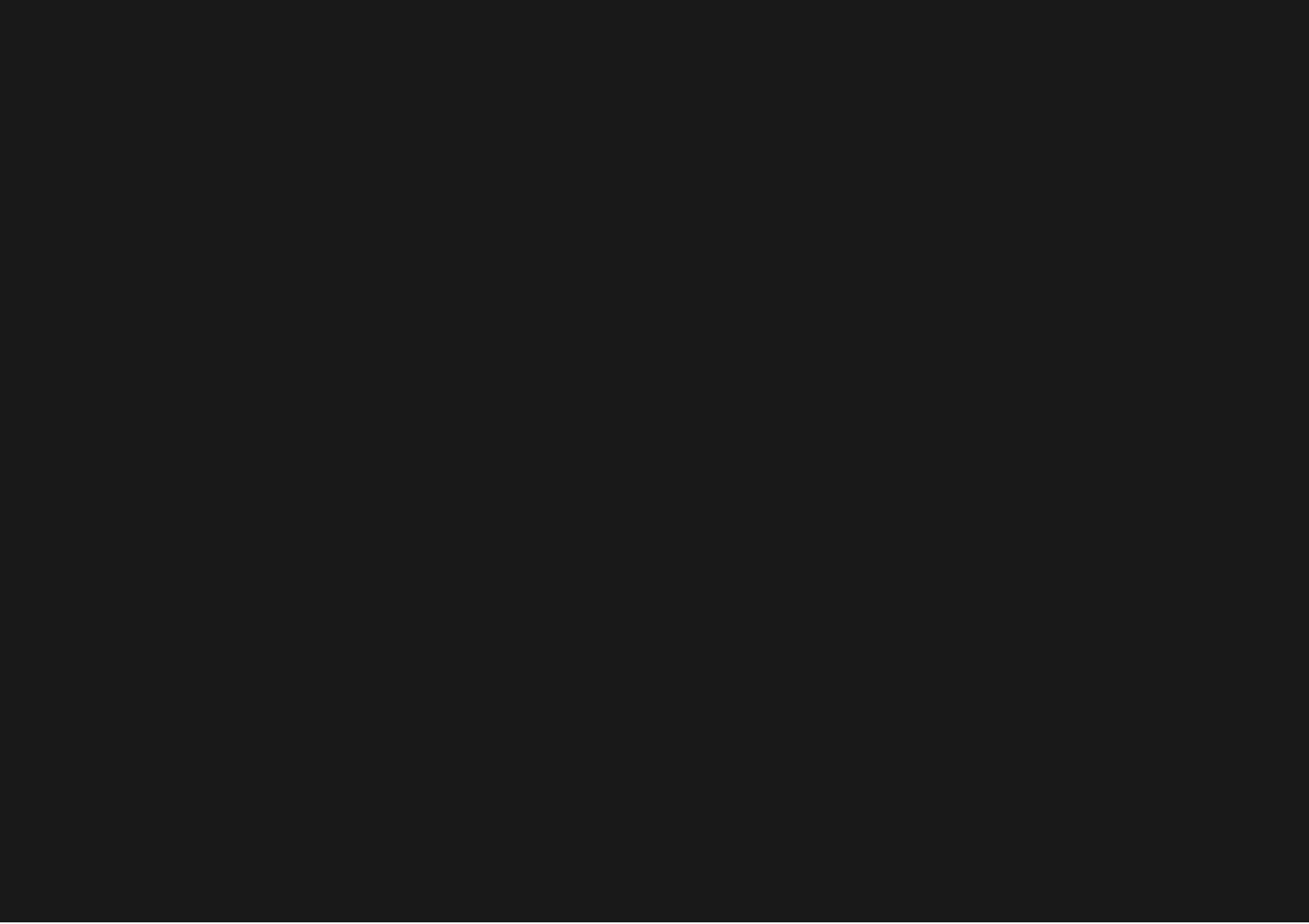
Rust 编程原理与实践





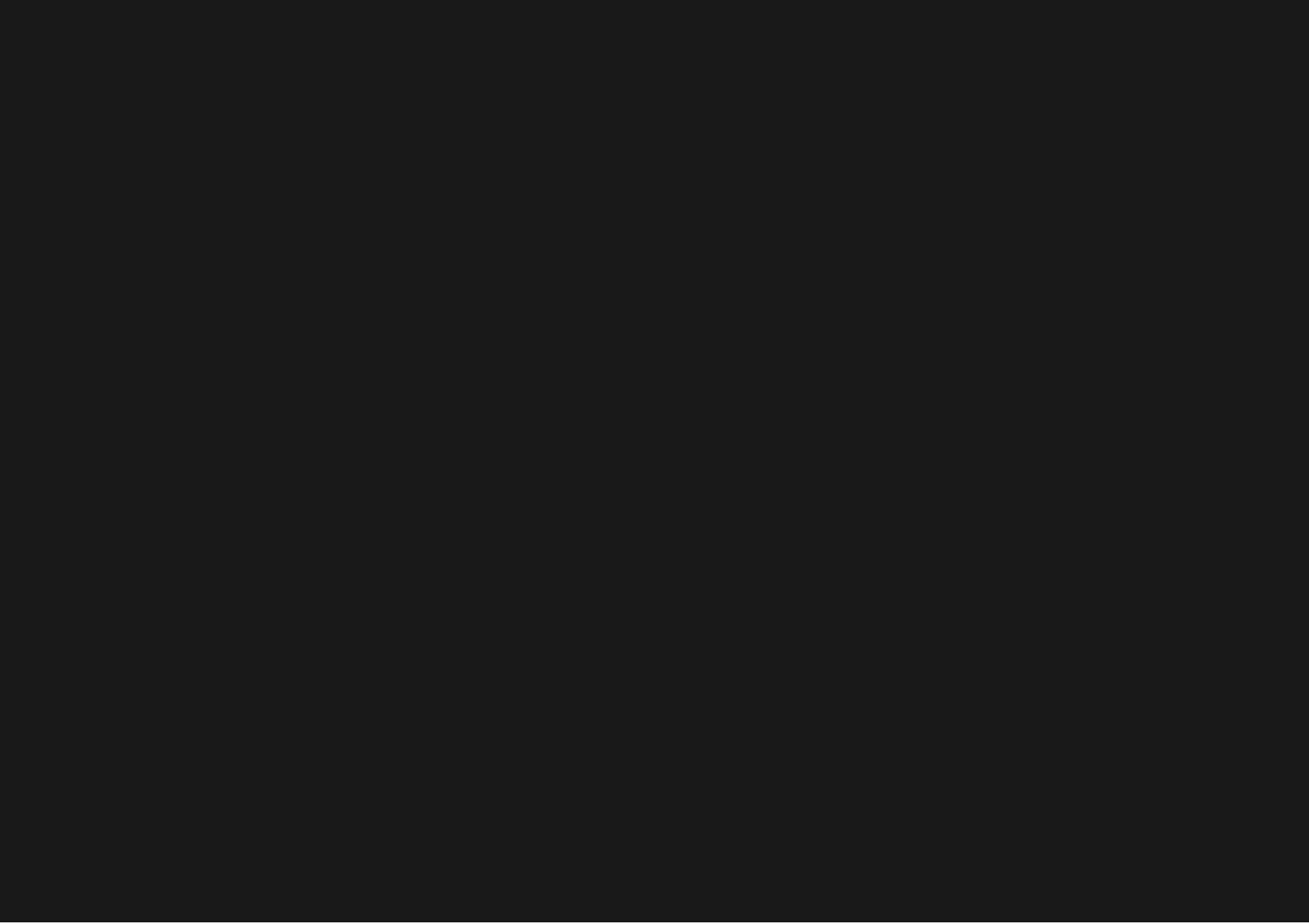
# PROGRAMMING LANGUAGE THEORY

- SICP (MIT 6.001) and Computer Magic
- Functional programming: Haskell, Elixir,  $\lambda$ -calculus...
- Formal Proof: proof assistant  $\rightarrow$  Agda, Arend...
- Type Theory: HoTT, CuTT, Dependent type  $\rightarrow$  Agda, Idris...
- Implement a lisp/scheme interpreter using Haskell...
- Write a C compiler using Rust  $\rightarrow$  ref: rcc
- Design your own programming language



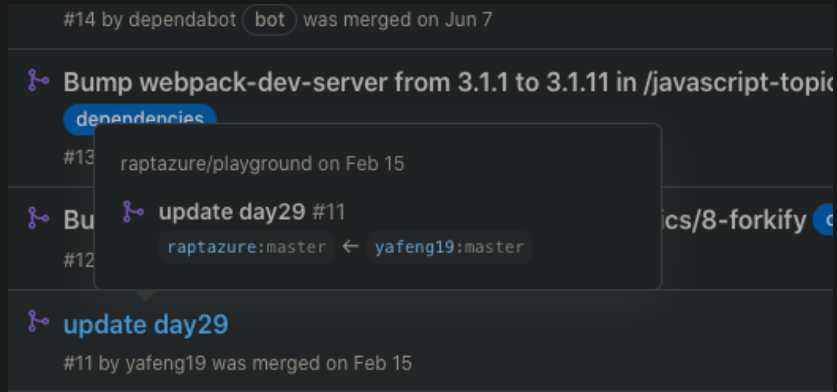
## II. ORGANIZE YOUR TEAM

- Open Source Workflow
- GitHub Project



# OPEN SOURCE WORKFLOW

- Push your code to GitHub (using git)
- Pull Requests / Code Review
- Merge PR and work together
- Assistant: QQ Group





# GITHUB PROJECT

- Manage your progress
- Assign tasks to your team members

The screenshot displays a GitHub Project board with four columns: "To do", "In progress", "Done", and "Noticed".

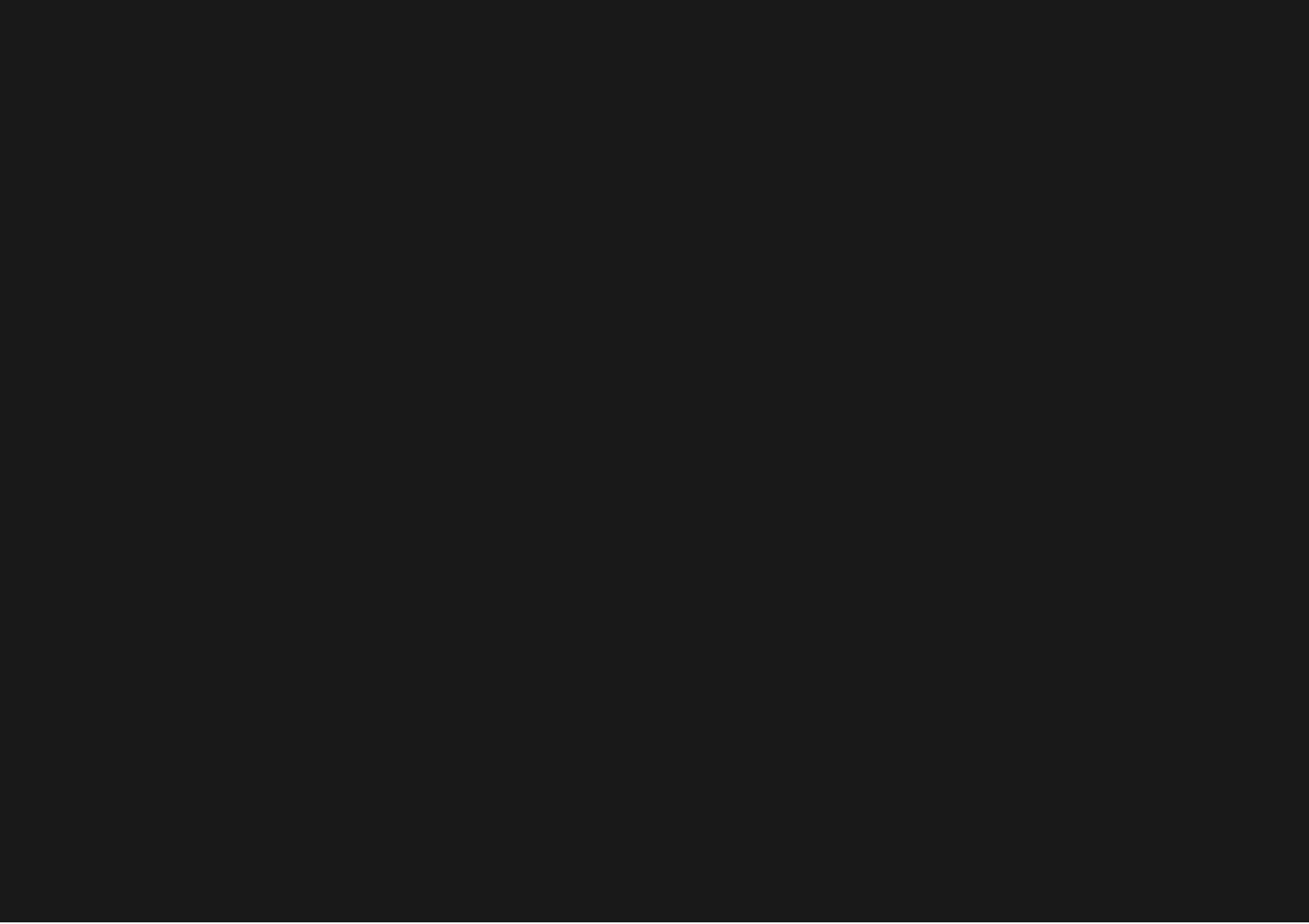
- To do (4 items):**
  - Daily Project: key-value store
    - ☒ The Rust toolbox
    - ☒ Log-structured file I/O
    - ☐ Synchronous client-server networking
    - ☐ Concurrency and Parallelism
    - ☐ Asynchronous programming in RustAdded by raptazure
  - Daily Project: raft k-v store
    - ☐ The Percolator lab
    - ☐ The Raft labAdded by raptazure
  - Daily Project: simple redis
    - ☐ Spawning
    - ☐ Shared state
    - ☐ Channels
    - ☐ I/O
    - ☐ Framing
    - ☐ Async in depth
    - ☐ Select
    - ☐ StreamsAdded by raptazureAutomated as To do Manage
- In progress (3 items):**
  - Daily Project: many linked lists
    - ☒ A bad safe deque
    - ☐ An unsafe queue
    - ☐ The double single listAdded by raptazure
  - Daily Lesson: 6.824
    - ☒ RPC and Threads
      - ☐ GFS
      - ☐ Primary-backup replication
      - ☐ Threads and Raft
      - ☐ Fault Tolerance 1
      - ☐ Fault Tolerance 2
      - ☐ Zookeeper
      - ☐ More replication, CRAQ
      - ☐ Cloud replication DB
      - ☐ Cache consistency
      - ☐ Distributed Transactions
      - ☐ Spanner
      - ☐ Optimistic Concurrency Control
      - ☐ Big Data: Spark
      - ☐ Memcached
      - ☐ COPS, Causal Consistency
      - ☐ Fork ConsistencyAutomated as In progress Manage
- Done (1 item):**
  - Daily Project: browser engine
    - ☒ Build the DOM and the HTML Parser.
    - ☒ Add a CSS engine and CSS parsing.
    - ☒ Add a Style Tree.
    - ☒ Boxes and the Layout Tree.
    - ☒ Commands and Rendering in OpenGL.
    - ☒ Finish up.Added by raptazureAutomated as Done Manage
- Noticed (1 item):**
  - Added by raptazure
    - coprocessor/expression: push down scalar functions
    - 333 of 469
    - #3275 opened by AndreMouche in tikv/tikv
    - difficulty/easy sig/coprocessor
    - status/help-wanted





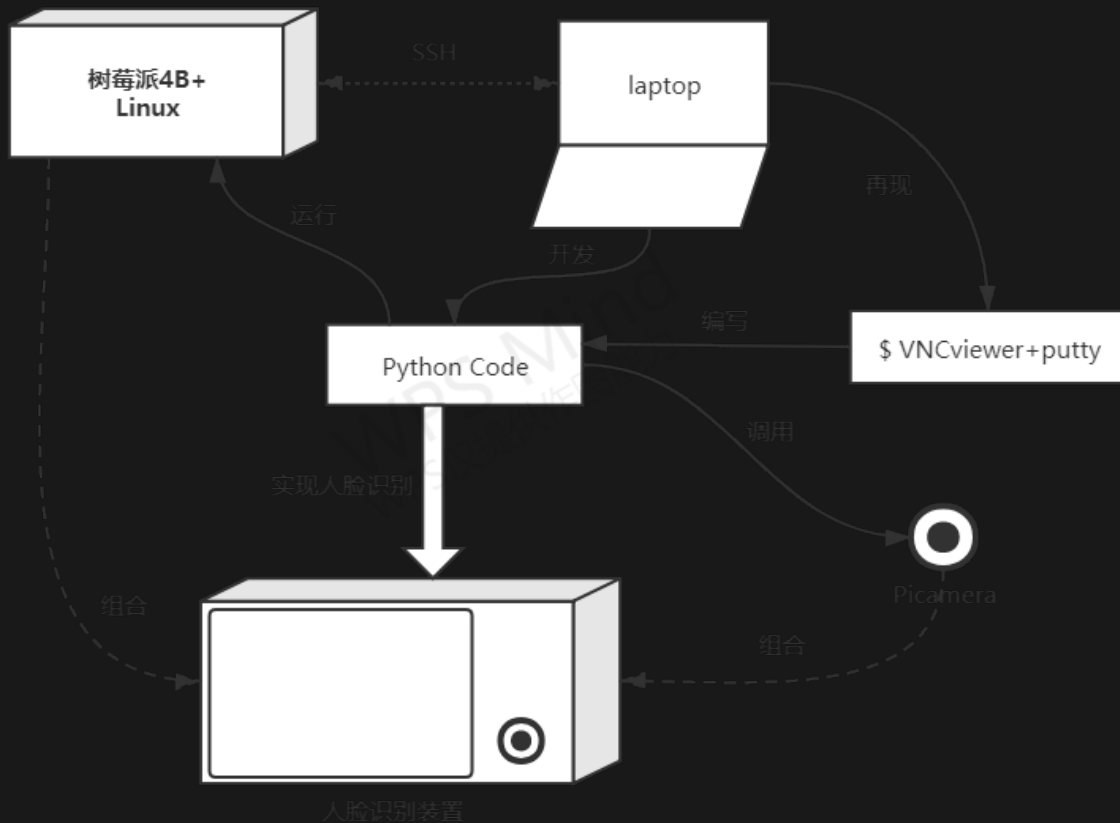
# III. PRESENT YOUR IDEAS

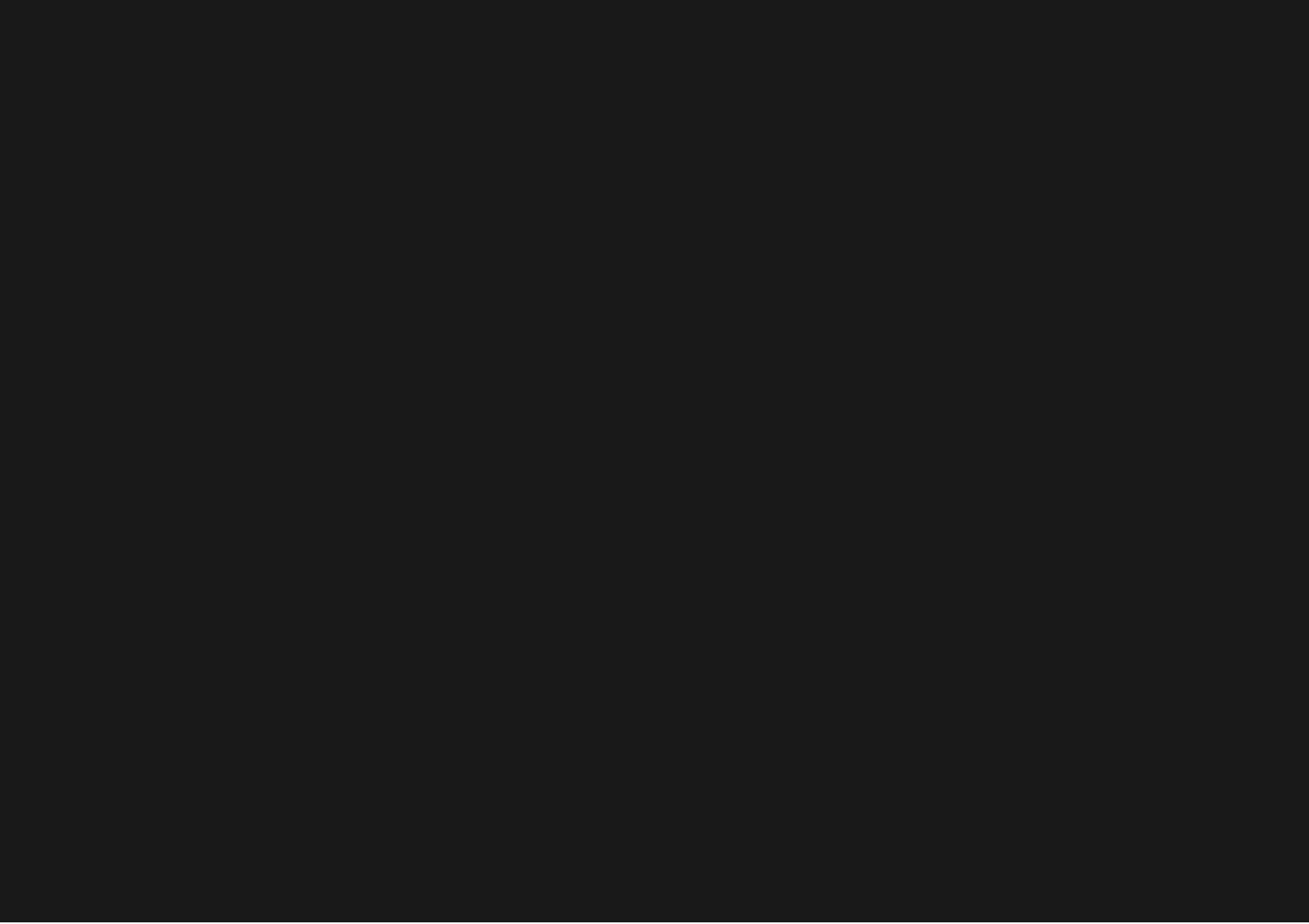
- Creation and Presentation



# CREATION AND PRESENTATION

- Everything creative: from your thoughts to the implementation...
- Show it to others: module graph & flowchart





# THANK YOU!

- QQ: 1051276278
- <https://github.com/raptazure>

