SELECTED TECHNICAL WORK

SOFTWARE ENGINEERING PORTFOLIO

Cross-Language Interactive Systems Suite

Matthew Moncada

Full-Stack Web Developer | Software Engineering | Digital Systems

C++ | Java | JavaScript | Python | HTML/CSS

5	2,357	4	3
APPLICATIONS	SOURCE LINES	PROGRAMMING LANGUAGES	INTERFACE PARADIGMS

PORTFOLIO THESIS

A set of educational game-engine prototypes that demonstrates the ability to translate the same domain - player state, combat, inventory, progression, validation, and navigation - across native command-line, desktop GUI, and browser runtimes.

CONTACT

Colorado-based | (210) 322-4787 | matthewmoncada90@gmail.com

linkedin.com/in/matthew-moncada | github.com/mattcodingsensei

MM

PROFESSIONAL POSITIONING

Software engineer with a production web background and demonstrable cross-language fundamentals.

This portfolio isolates the engineering capability behind Matthew Moncada's broader web-development career. The featured applications were created as instructional projects for teaching programming concepts, but the source code also documents practical experience with object-oriented design, application state, algorithms, event-driven interfaces, asynchronous control flow, file I/O, and modular architecture.

01 Systems Modeling

Represented player, enemy, world, inventory, economy, and progression state as interacting software entities.

02 Cross-Runtime Translation

Implemented comparable domain logic across C++, Java, JavaScript, Python, desktop, browser, and terminal environments.

03 Interface Engineering

Built command-driven flows, Swing event handlers, dynamic DOM controls, and Promise-based browser input.

04 Instructional Design

Structured the applications to expose core concepts through visible state transitions, choices, combat, and progression.

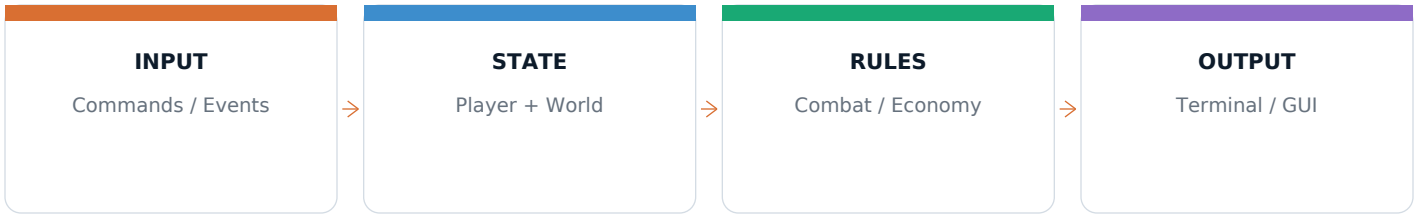
ENGINEERING SIGNAL

The value is not the game genre. It is the repeatable ability to model a problem, define state and rules, route user actions, calculate outcomes, and render the result in the conventions of each platform.

matthewmoncada90@gmail.com | github.com/mattcodingsensei

Cross-Language Systems Suite

Five implementations. One shared engineering problem space.



PROJECT	LANGUAGE	RUNTIME	ARCHITECTURE	CORE SYSTEMS
Space Adventure RPG	Python	Terminal	5 modules / 4 domain classes	Procedural worlds, combat, XP, economy, galaxy gates
Medieval Adventure	Java	Swing desktop	Event-driven class methods	GUI state, listeners, combat, branching scenes
Medieval GUI	JavaScript	Browser	ES6 class + DOM	Dynamic controls, inventory, rendering, callbacks
Terminal Adventure	JavaScript	Browser terminal	async/await + Promises	Event-to-Promise input, validation, command routing
Medieval Terminal	C++	Native CLI	Class methods + streams	File I/O, arrays, validation, combat, persistent state

SOURCE REVIEW SNAPSHOT

13 code files **2,357 lines** **6 technologies**

Reviewed across C++, Java, JavaScript, Python, HTML, and CSS source files.

matthewmoncada90@gmail.com | github.com/mattcodingsensei

Python Space Adventure RPG

Modular domain model, procedural generation, and layered progression.

```
What would you like to do next?
travel
Finding a planet...
Planet located!
This planet's name is Pohz
This region's climate is Cold
You can find 6 hectoliters of fuel on this planet
You can find 34 megagrams of resources on the planet
This planet's tier level is 1
Do you want to travel to this planet? [y | n]y
Traveling to Pohz...
What would you like to do at Pohz? [mine | rest | shop | leave]mine
You went out to mine for resources...
 Congrats! You found 34 resources and 6 fuel!
You are satisfied and have left the planet.
What would you like to do next?
```

APPLICATION SCOPE

The largest application in the suite: 967 lines across five Python modules.

- Player, Alien, Boss, and Planet domain classes
- Randomized names, climates, fuel, resources, and enemy statistics
- Turn-based attack, defense, parry, escape, and boss encounters
- Experience, leveling, health, attack, defense, and currency progression
- Resource-gated advancement across multiple galaxies

MODULE ARCHITECTURE

main.py

Orchestration, command routing, travel, economy, combat loops

player.py

Player state, attack, defense, parry, XP, leveling, money

alien.py

Procedural names, stat generation, randomized combat behavior

boss.py

Specialized boss entity and encounter behavior

planet.py

Planet generation, environment data, resource tiers

player.py - stateful combat behavior

```
class Player:
  def player_parry(self, alien):
    if alien.current_move == "attack":
      alien.defense /= 2
      damage = self.player_attack(alien)
      alien.defense *= 2
      alien.current_move = "parried"
```

ENGINEERING EVIDENCE

- Separation of domain behavior into imported modules rather than a single script.
- Procedural content generation using randomized data sets and tier-aware rules.
- Nested game loops coordinating commands, encounters, rewards, and fail states.
- Progression algorithms connecting experience, character statistics, currency, and resource thresholds.

matthewmoncada90@gmail.com | github.com/mattcodingsensei

The same game domain implemented through two different runtime models.

Java Swing Desktop Adventure

445 lines | Swing + AWT | Event-driven GUI

- Constructed a native desktop interface with JFrame, JPanel, JButton, JLabel, JTextArea, and GridLayout.
- Implemented ActionListener handlers that dispatch user actions based on current application position.
- Updated narrative text, labels, button commands, health, weapon state, and component visibility dynamically.
- Modeled branching scenes and win/loss transitions as state-dependent methods.
- Calculated randomized combat damage based on weapon selection and current health.

Event-driven command dispatch

```
public void actionPerformed(ActionEvent event) {
  String choice = event.getActionCommand();
  switch(position) {
    case "townGate": ...
    case "crossRoad": ...
    case "fight": ...
  }
}
```

C++ Terminal Adventure

364 lines | Native CLI | Streams + file I/O

- Encapsulated scene logic in a Game class with methods for setup, navigation, shops, combat, and ending states.
- Persisted an introduction through ofstream and reloaded it through ifstream and getline.
- Managed player health, gold, weapon, potion arrays, enemy health, and quest state across the application.
- Used stream failure recovery to validate numeric input and return the user to the correct command flow.
- Implemented randomized weapon damage, inventory changes, combat outcomes, and conditional victory.

Input validation and recovery

```
if (cin.fail()) {
  cin.clear();
  cin.ignore(numeric_limits<streamsize>::max(),
             '\n');
  cout << "Please enter a number";
  crossRoads();
}
```

CROSS-LANGUAGE TRANSFER

SHARED CONCEPT	JAVA IMPLEMENTATION	C++ IMPLEMENTATION
User action	Swing ActionEvent / button commands	cin command selection
Application state	position + component state	global state + Game methods
Output	JTextArea / JLabel updates	cout terminal output
Validation	action-command branching	stream fail recovery
Persistence	in-memory desktop session	fstream introduction file

Two interfaces: a dynamic GUI and an asynchronous terminal experience.

JavaScript Browser GUI

ES6 class | DOM manipulation | data-driven actions | 236 JavaScript lines

- Central Game class stores player health, currency, weapons, inventory, enemy state, identity, and quest completion.
- Reusable updateStats(), showMessage(), and showChoices() methods separate rendering from game actions.
- Choice buttons are created from action objects, bound with addEventListener(), and inserted into the DOM dynamically.
- Inventory operations use arrays, includes(), push(), filter(), and conditional validation before state changes.
- The browser interface re-renders available actions after every state transition.

Data-driven DOM rendering

```
showChoices(choices) {
  const container = document.getElementById(
    "game-choices");
  container.innerHTML = "";
  choices.forEach((choice) => {
    const button = document.createElement("button");
    button.addEventListener("click", choice.action);
    container.appendChild(button);
  });
}
```

JavaScript Browser Terminal

async/await | Promises | form events | regex validation | 235 JavaScript lines

Event-to-Promise input adapter

```
getInput(promptText) {
  this.print(promptText);
  return new Promise((resolve) => {
    this.awaitingInput = resolve;
  });
}

const choice = await this.getInput("> ");
```

- Wrapped browser form submissions in Promises so event-driven input could be consumed through sequential async/await game logic.
- Captured submit events with preventDefault(), sanitized values, cleared the input, and resolved the pending callback.
- Validated player names with length checks and a regular expression before advancing application state.
- Used switch-based command routing for town, crossroads, shop, exploration, combat, and terminal failure states.
- Maintained Git history documenting the refactor to a browser-compatible runtime.

Repositories: github.com/mattcodingsensei/JavaScript-Medieval-Game-with-Web-Browser-GUI
github.com/mattcodingsensei/JavaScript-Terminal-Game

Engineering Competency Matrix

Core computer-science fundamentals reinforced by production web and infrastructure experience.

DEMONSTRATED DIRECTLY IN THE SOURCE CODE

Object-Oriented Design
Classes, methods, encapsulated behavior, interacting domain entities

State Management
Health, inventory, currency, equipment, progression, enemies, locations

Algorithms & Rules
Damage calculation, randomization, leveling, thresholds, win/loss logic

Event-Driven Programming
Swing listeners, DOM events, dynamic button callbacks

Asynchronous JavaScript
Promises, async/await, browser form-to-input orchestration

Procedural Generation
Alien names, planet data, climates, resources, enemy statistics

Input Validation
Regular expressions, stream recovery, menu guards, command routing

File & Stream I/O
C++ fstream, getline, Python stdout flushing, terminal interactions

Version Control
Git repositories and iterative commit history

ADDITIONAL PROFESSIONAL ENGINEERING CAPABILITIES

Frontend Engineering
React, JavaScript ES6+, HTML5, CSS3/SCSS, Tailwind CSS, Astro, responsive UI, cross-browser QA

CMS & Platform Development
WordPress, Gutenberg, Advanced Custom Fields, custom themes, plugins, PHP, Shopify Liquid, Elementor, Divi

APIs & Integrations
REST APIs, Google APIs, Google Maps API, Zillow API, third-party scheduling, analytics, marketing and form integrations

Local Development
MAMP, VVV, Docker, npm, local-vs-hosted environment configuration, dependency and build workflows

Hosting & Web Servers
Apache, Nginx, Linux/Bash, Nano, file permissions, redirects, deployment troubleshooting, backups and migrations

DNS, Email & Security
A/AAAA/CNAME/TXT records, MX email records, SPF/DKIM/DMARC, SSL/TLS, domain routing, production incident resolution

Analytics & Optimization
GA4, Google Tag Manager, Search Console, Meta Pixel, conversion events, KPI measurement, Core Web Vitals, technical SEO

Delivery & Operations
Git/GitHub, branching and merging, hosting migrations, launch validation, documentation, stakeholder communication

matthewmoncada90@gmail.com | github.com/mattcodingsensei

A web-platform professional with evidence of deeper software engineering fundamentals.

WHY THIS WORK MATTERS

These projects demonstrate an engineering foundation that supports Matthew's professional specialization in web platforms, analytics, hosting, and digital systems. Across each implementation, the core work is the same: decompose a problem, model state, define rules, manage input, coordinate control flow, and render a reliable interface.

That foundation transfers directly to product engineering, frontend development, API integration, platform operations, and technical debugging across the application stack.

A Builds Across Layers

Can move from algorithms and domain state to user interfaces, APIs, hosting, analytics, and production operations.

B Learns by Building

Has repeatedly translated concepts into working code rather than relying only on platform-level configuration.

C Communicates Systems

Created the projects as instructional material, requiring technical concepts to be organized into teachable workflows.

CREDENTIALS


Certificate of Accomplishment

Software Engineer Intern

PRESENTED TO

Matthew Moncada

The bearer of this certification has passed Hackerrank role certification exam

Earned on: 13 Mar, 2024

ID: 036B2D171207

Harishankaran K

CTO, HackerRank

Education & Certifications

- Associate of Science in Computer Science - Community College of Denver, 2024
- HackerRank Software Engineer Intern role certification - earned March 13, 2024
- HTML5, CSS3, and JavaScript certifications
- 7+ years of web development and digital platform experience

MATTHEW MONCADA Colorado | (210) 322-4787 | matthewmoncada90@gmail.com
linkedin.com/in/matthew-moncada | github.com/mattcodingsensei