4.4 Studio Development Roadmap Ver 2.3

# Table of Contents

[0 Table of Contents 1](#_Toc181699847)

[1 Project Foundations 6](#_Toc181699848)

[1.1 Project Overview 6](#_Toc181699849)

[1.2 Vision Statement 6](#_Toc181699850)

[1.3 Studio Development Philosophy 6](#_Toc181699851)

[1.4 Meta-Game Foundational Principles 6](#_Toc181699852)

[2 Master Design Document 6](#_Toc181699853)

[2.1 Genre Influences 6](#_Toc181699854)

[2.2 Gameplay Influences List 6](#_Toc181699855)

[2.3 High-Level Objectives 6](#_Toc181699856)

[2.4 Gameplay Pillars List 6](#_Toc181699857)

[2.5 Universal Gameplay Loops List 6](#_Toc181699858)

[2.6 Simulation Progression Plan 7](#_Toc181699859)

[2.7 Graphical Progression Plan 7](#_Toc181699860)

[2.8 Player Profile Progression Plan 7](#_Toc181699861)

[3 Master Technical Design Document 7](#_Toc181699862)

[3.1 Project Setup and Organization 7](#_Toc181699863)

[3.1.1 Version Control System (VCS) Setup 7](#_Toc181699864)

[3.1.2 Project Structure and Naming Conventions 7](#_Toc181699865)

[3.1.3 Unity Project Settings 7](#_Toc181699866)

[3.2 Core ECS/DOTs Architecture 7](#_Toc181699867)

[3.2.1 ECS Foundations 7](#_Toc181699868)

[3.2.2 Component Design 7](#_Toc181699869)

[3.2.3 System Implementation 7](#_Toc181699870)

[3.2.4 Entity Management 7](#_Toc181699871)

[3.2.5 Composition Over Inheritance 7](#_Toc181699872)

[3.3 Modular and Scalable Design 7](#_Toc181699873)

[3.3.1 Feature Modules 7](#_Toc181699874)

[3.3.2 Event System Implementation 7](#_Toc181699875)

[3.3.3 Data-Driven Configuration 7](#_Toc181699876)

[3.3.4 Plug-In Architecture 7](#_Toc181699877)

[3.4 Rendering and Visuals 7](#_Toc181699878)

[3.4.1 Hybrid Renderer Setup 7](#_Toc181699879)

[3.4.2 Art Asset Management 7](#_Toc181699880)

[3.4.3 Animation Integration 7](#_Toc181699881)

[3.4.4 VFX and Shader Management 7](#_Toc181699882)

[3.5 UI and User Interaction 7](#_Toc181699883)

[3.5.1 UI Framework Selection 7](#_Toc181699884)

[3.5.2 Data Synchronization Between ECS and UI 7](#_Toc181699885)

[3.5.3 Input Handling 7](#_Toc181699886)

[3.5.4 Responsive and Adaptive UI Design 7](#_Toc181699887)

[3.6 Resource and Scene Management 7](#_Toc181699888)

[3.6.1 Asset Loading with Addressables 7](#_Toc181699889)

[3.6.2 Scene Organization 7](#_Toc181699890)

[3.6.3 World Streaming System 7](#_Toc181699891)

[3.6.4 Resource Cleanup 7](#_Toc181699892)

[3.7 AI and NPC Behavior 7](#_Toc181699893)

[3.7.1 AI Architecture Planning 7](#_Toc181699894)

[3.7.2 Modular AI Components 7](#_Toc181699895)

[3.7.3 Pathfinding and Navigation 7](#_Toc181699896)

[3.7.4 Performance Optimization 7](#_Toc181699897)

[3.8 Testing and Quality Assurance 7](#_Toc181699898)

[3.8.1 Automated Testing Framework 7](#_Toc181699899)

[3.8.2 Continuous Integration (CI) 7](#_Toc181699900)

[3.8.3 Debugging Tools 7](#_Toc181699901)

[3.8.4 Error Handling and Logging 7](#_Toc181699902)

[3.9 Performance Profiling and Optimization 7](#_Toc181699903)

[3.9.1 Regular Profiling Sessions 7](#_Toc181699904)

[3.9.2 Burst Compiler Utilization 7](#_Toc181699905)

[3.9.3 Job System Optimization 7](#_Toc181699906)

[3.9.4 Memory Management 7](#_Toc181699907)

[3.10 Documentation and Code Maintenance 7](#_Toc181699908)

[3.10.1 Comprehensive Documentation 7](#_Toc181699909)

[3.10.2 Code Reviews and Standards 7](#_Toc181699910)

[3.10.3 Technical Debt Management 7](#_Toc181699911)

[3.10.4 Knowledge Sharing 7](#_Toc181699912)

[3.11 Analytics and Player Feedback 7](#_Toc181699913)

[3.11.1 Analytics Integration 7](#_Toc181699914)

[3.11.2 Feedback Systems 7](#_Toc181699915)

[3.11.3 Data Privacy Compliance 7](#_Toc181699916)

[3.12 Planning for Scalability and Future Expansions 7](#_Toc181699917)

[3.12.1 Scalability Testing 7](#_Toc181699918)

[3.12.2 Cross-Platform Considerations 7](#_Toc181699919)

[3.12.3 Modding Support 7](#_Toc181699920)

[3.12.4 Long-Term Maintenance Strategy 7](#_Toc181699921)

[3.13 Legal and Licensing 7](#_Toc181699922)

[3.13.1 Asset Licensing Compliance 7](#_Toc181699923)

[3.13.2 Open-Source Components 7](#_Toc181699924)

[3.13.3 Trademarks and Branding 7](#_Toc181699925)

[3.14 Team Collaboration Tools 7](#_Toc181699926)

[3.14.1 Communication Platforms 7](#_Toc181699927)

[3.14.2 Project Management Software 7](#_Toc181699928)

[3.14.3 Asset Management 7](#_Toc181699929)

[4 Expansion Roadmap (Expansion->Milestones->Feature Groups) 7](#_Toc181699930)

[4.1 Text to Graphics 7](#_Toc181699931)

[4.1.1 Milestone 1: 7](#_Toc181699932)

[4.1.2 Milestone 2: 7](#_Toc181699933)

[4.1.3 Milestone 3: 7](#_Toc181699934)

[4.1.4 Milestone 4: 7](#_Toc181699935)

[4.2 7](#_Toc181699936)

[4.2.1 Milestone 1: 7](#_Toc181699937)

[4.2.2 Milestone 2: 7](#_Toc181699938)

[4.2.3 Milestone 3: 7](#_Toc181699939)

[4.2.4 Milestone 4: 7](#_Toc181699940)

[4.3 7](#_Toc181699941)

[4.3.1 Milestone 1: 7](#_Toc181699942)

[4.3.2 Milestone 2: 7](#_Toc181699943)

[4.3.3 Milestone 3: 7](#_Toc181699944)

[4.3.4 Milestone 4: 7](#_Toc181699945)

[4.4 7](#_Toc181699946)

[4.4.1 Milestone 1: 7](#_Toc181699947)

[4.4.2 Milestone 2: 7](#_Toc181699948)

[4.4.3 Milestone 3: 7](#_Toc181699949)

[4.4.4 Milestone 4: 7](#_Toc181699950)

[4.5 7](#_Toc181699951)

[4.5.1 Milestone 1: 7](#_Toc181699952)

[4.5.2 Milestone 2: 7](#_Toc181699953)

[4.5.3 Milestone 3: 7](#_Toc181699954)

[4.5.4 Milestone 4: 7](#_Toc181699955)

[4.6 7](#_Toc181699956)

[4.6.1 Milestone 1: 7](#_Toc181699957)

[4.6.2 Milestone 2: 7](#_Toc181699958)

[4.6.3 Milestone 3: 7](#_Toc181699959)

[4.6.4 Milestone 4: 7](#_Toc181699960)

[4.7 7](#_Toc181699961)

[4.7.1 Milestone 1: 7](#_Toc181699962)

[4.7.2 Milestone 2: 7](#_Toc181699963)

[4.7.3 Milestone 3: 7](#_Toc181699964)

[4.7.4 Milestone 4: 7](#_Toc181699965)

[4.8 7](#_Toc181699966)

[4.8.1 Milestone 1: 7](#_Toc181699967)

[4.8.2 Milestone 2: 7](#_Toc181699968)

[4.8.3 Milestone 3: 7](#_Toc181699969)

[4.8.4 Milestone 4: 7](#_Toc181699970)

[4.9 7](#_Toc181699971)

[4.9.1 Milestone 1: 7](#_Toc181699972)

[4.9.2 Milestone 2: 7](#_Toc181699973)

[4.9.3 Milestone 3: 7](#_Toc181699974)

[4.9.4 Milestone 4: 7](#_Toc181699975)

[4.10 7](#_Toc181699976)

[4.10.1 Milestone 1: 7](#_Toc181699977)

[4.10.2 Milestone 2: 7](#_Toc181699978)

[4.10.3 Milestone 3: 7](#_Toc181699979)

[4.10.4 Milestone 4: 7](#_Toc181699980)

[4.11 End 7](#_Toc181699981)

[5 Universe Bible 7](#_Toc181699982)

[6 Art Bible 7](#_Toc181699983)

[7 Live Operations Plan 7](#_Toc181699984)

[8 End 7](#_Toc181699985)

# Project Foundations

## Project Overview

* Project Codename: SimCraft RPG
* Studio Assigned: Stoned Logic Games
* Primary Genre: Sandbox Strategy RPG

## Vision Statement

* An epic, multi-genre game that blends RPG character development, massive NPC simulation, 4X strategy, city-building, and resource management with intricate narrative elements. Set in a large 2D top-down open world, players take on dual roles: as a single character navigating quests, factions, and world events, and as the leader of an expansive organization or faction overseeing up to 100,000 NPCs. Players must balance diplomacy, economy, military strategy, and cultural influence, while building cities, managing resources, and guiding the lives of NPCs who have their own goals, relationships, and personal arcs.

## Studio Development Philosophy

* Guiding Principle 1:  
  Develop a live-service type game over the course of the life of the studio. Start with the smallest possible MVP and continue to scale up features though expansions to result in all the studio’s effort resulting into one interconnected product.
* Guiding Principle 2:  
  Develop game making tools underneath the Unity engine framework that will allow for faster game expansions over time as the studio toolkit, codebase, and asset libraries expands. Use automated tools to increase asset production speeds as a solo, indie developer.
* Guiding Principle 3:  
  Use game and technical design principles to develop building blocks of gameplay that allow for emergent outcomes when the blocks are combined differently in a sandbox environment.
* Guiding Principle 4:  
  Build a simulation universe in a realistic way rather than a gamified way first. Only move off realism if prototype feedback demands changes to more typical gamification paths. There should always be unseen activity that the player is not seeing to maintain the consistency of the progression of the simulated world.

## Meta-Game Foundational Principles

* Foundational Principle 1:  
  Players can think strategically about what they are doing without feeling time pressure. Mechanics will not be dependent on physical speed or reaction time.
* Foundational Principle 2:  
  The player should never feel like they have wasted time playing because they have bad outcomes. Everything step should bring a constant sense of progression and trade-offs whether with success or failure. Reward players with positives they didn’t foresee coming when plans do not work out as envisioned, just as happens in real life-silver linings etc.
* Foundational Principle 3:  
  All features should start with the most popular core mechanics that are fundamentally fun to play by themselves. The real nature of a particular feature should emerge as logical accessory mechanics are added to the core mechanic(s) leading to a best practice feature package.
* Foundational Principle 4:  
  Reduce decision fatigue and “forgetting where you left off” as barriers of entry to player jumping into the game for shorter play sessions. Explore the ideas of having decisions presented when immersive as much as possible without limiting the space of possibilities.

# Master Design Document

## Genre Influences

### Text-Based Games

### End

## Gameplay Influences List

### Zork

* Genre: Text Adventure
* Major Aspect(s): Text-Only Gameplay
* Minor Key Aspects: Room Descriptions; Text Parser

## High-Level Objectives

### Technical Ramp-Up

* Use an immersive narrative from the opening screen of the game to introduce the idea that the player can only see text at first. In fact, they can only read text and cannot even select anything. Over time, the narrative justifies why the player experiences a stream of technological improvements to the game in a incremental and iterative way.

## Gameplay Pillars List

### Immersive Narrative

* Present the player that explains to the player “in-narrative” why everything on the screen exists including things typically left unexplained like UI elements and menus. To do this, we will use the story-within-a-story technique where there is meta-narrative explaining why the experience the player is having looks like a game.

## Universal Gameplay Loops List

* Observe->Learn->Develop->Implement->Manage->Upgrade

## Simulation Progression Plan

### Competition Amongst Prisoners

* The opening of the game lays out the player is a prisoner competing against other prisoners for a chance at freedom. The simulation aspect of the game should start with creating the NPC prisoner pool that the player will begin competing against very near the start. Some of these NPC prisoners will persist until the very end of the game.

## Graphical Progression Plan

### Reading Only

* The opening of the game involves the player only able to read text without any input commands. The camera will not even be able to be moved by the game. Small incremental improvements starting with the ability for the “operator” of the game to start scrolling the text and continuing the digital novel section.

## Player Profile Progression Plan

### The Brain

* The player will not realize at first but the text is not being removed from the background as the camera scrolls. Eventually the player will realize that the black screen space they are currently seeing is a very rudimentary representation of their brain with text, arrayed across the screen like many pages of book, representing possible thoughts.

# Master Technical Design Document

## Project Setup and Organization

### Version Control System (VCS) Setup

* Initialize a Git repository for your project.
* Create a .gitignore file to exclude unnecessary files (e.g., Library/, Temp/, Build/ folders).
* Decide on a branching strategy (e.g., GitFlow, feature branches).

### Project Structure and Naming Conventions

* Establish a clear folder hierarchy (e.g., Assets/Scripts/, Assets/Art/, Assets/Scenes/).
* Define and document naming conventions for files, folders, components, systems, and assets.
* Use namespaces to organize code and prevent naming conflicts.

### Unity Project Settings

* Configure player settings for target platforms.
* Set up quality settings suitable for development (can be adjusted later for production).
* Adjust Time settings if necessary (fixed timestep, maximum delta time).

## Core ECS/DOTs Architecture

### ECS Foundations

* Familiarize yourself with Unity's ECS concepts: Entities, Components (IComponentData), and Systems (ISystem).
* Install the latest ECS, Burst Compiler, and Job System packages compatible with Unity 6.

### Component Design

* Define data-only components for your game (e.g., HealthComponent, PositionComponent).
* Avoid adding behavior or methods to components; keep them as pure data containers.

### System Implementation

* Create systems to process entities with specific components (e.g., MovementSystem, CombatSystem).
* Use the Job System and Burst Compiler to optimize system performance where applicable.

### Entity Management

* Decide on an entity spawning and destruction strategy.
* Use entity prefabs or archetypes to standardize entity creation.

### Composition Over Inheritance

* Favor composing entities with multiple components over using inheritance hierarchies.
* Design entities to be flexible by adding or removing components as needed.

## Modular and Scalable Design

### Feature Modules

* Organize game features into separate modules or packages (e.g., Combat, AI, Inventory).
* Ensure modules are decoupled and communicate via events or shared components.

### Event System Implementation

* Create an event or messaging system within ECS to handle interactions between systems.
* Use event components or an event manager to publish and subscribe to events (e.g., DamageEvent, ItemCollectedEvent).

### Data-Driven Configuration

* Externalize game data into configuration files (JSON, XML, ScriptableObjects).
* Implement systems to load and apply configurations at runtime.

### Plug-In Architecture

* Design your game to support plugins or add-ons for future expansions.
* Ensure new modules can be integrated without modifying core systems.

## Rendering and Visuals

### Hybrid Renderer Setup

* Use Unity's Hybrid Renderer to link ECS data with rendering components.
* Attach RenderMesh and LocalToWorld components to entities for rendering.

### Art Asset Management

* Organize art assets with a consistent naming convention.
* Use LOD (Level of Detail) systems for models to optimize rendering at different zoom levels.

### Animation Integration

* Decide on an animation system compatible with ECS (e.g., Unity's DOTS animation package).
* Implement animation components and systems to handle entity animations.

### VFX and Shader Management

* Plan for visual effects components and systems (e.g., particle effects).
* Use Shader Graph or custom shaders optimized for ECS rendering.

## UI and User Interaction

### UI Framework Selection

* Choose a UI framework (e.g., Unity UI Toolkit, UGUI) suitable for your game.
* Keep UI logic separate from ECS, but synchronize data as needed.

### Data Synchronization Between ECS and UI

* Implement systems or managers to update UI elements based on ECS component data.
* Use events to trigger UI updates when data changes (e.g., health bars, inventory updates).

### Input Handling

* Use Unity's Input System package for cross-platform input management.
* Map input actions to ECS components or systems appropriately.

### Responsive and Adaptive UI Design

* Design UI layouts that adapt to different resolutions and aspect ratios.
* Prepare for different gameplay modes requiring unique UI elements.

## Resource and Scene Management

### Asset Loading with Addressables

* Use Unity's Addressable Assets System to handle dynamic asset loading.
* Organize assets into addressable groups for efficient management.

### Scene Organization

* Divide your game world into manageable scenes or sub-scenes.
* Use additive scene loading to stream parts of the world as needed.

### World Streaming System

* Implement a system to load and unload entities based on player proximity.
* Optimize memory usage by only keeping nearby entities active.

### Resource Cleanup

* Ensure assets and entities are properly unloaded or destroyed when no longer needed.
* Avoid memory leaks by managing references carefully.

## AI and NPC Behavior

### AI Architecture Planning

* Decide on an AI system (behavior trees, state machines) suitable for ECS.
* Design AI components (e.g., AIStateComponent, TargetComponent) to represent NPC behavior.

### Modular AI Components

* Break down AI behaviors into reusable components.
* Allow for easy extension or modification of NPC behavior by adding or removing components.

### Pathfinding and Navigation

* Implement a pathfinding system compatible with ECS (e.g., A\* algorithm).
* Use navigation meshes or grids optimized for ECS.

### Performance Optimization

* Profile AI systems to ensure they scale well with many entities.
* Use Burst Compiler and multithreading to optimize AI calculations.

## Testing and Quality Assurance

### Automated Testing Framework

* Set up unit tests for individual systems and components.
* Write integration tests to verify interactions between systems.

### Continuous Integration (CI)

* Implement a CI pipeline using services like GitHub Actions, GitLab CI, or Jenkins.
* Automate builds and test runs on code commits.

### Debugging Tools

* Utilize Unity's debugging and profiling tools to monitor performance.
* Create custom editor windows or gizmos to visualize ECS data in the editor.

### Error Handling and Logging

* Implement robust error handling within systems.
* Use logging to track system behavior and catch issues early.

## Performance Profiling and Optimization

### Regular Profiling Sessions

* Schedule routine performance profiling to identify bottlenecks.
* Use Unity's Profiler and DOTS-specific profiling tools.

### Burst Compiler Utilization

* Apply the Burst Compiler to performance-critical systems.
* Ensure code is compatible with Burst (e.g., avoid managed types).

### Job System Optimization

* Leverage Unity's Job System for parallel processing.
* Minimize main thread usage to improve frame rates.

### Memory Management

* Monitor memory allocations to prevent excessive garbage collection.
* Use NativeArray and other native containers wisely.

## Documentation and Code Maintenance

### Comprehensive Documentation

* Document systems, components, and their interactions.
* Keep API documentation up to date for future reference.

### Code Reviews and Standards

* Establish code review practices to maintain code quality.
* Adhere to coding standards and best practices.

### Technical Debt Management

* Regularly refactor code to improve readability and performance.
* Track technical debt and prioritize its resolution.

### Knowledge Sharing

* Maintain a knowledge base or wiki for project-related information.
* Encourage collaboration and information sharing among team members.

## Analytics and Player Feedback

### Analytics Integration

* Incorporate analytics tools (e.g., Unity Analytics, Google Analytics) to collect player data.
* Track key metrics like player progression, session duration, and feature usage.

### Feedback Systems

* Implement in-game feedback forms or bug reporting tools.
* Encourage players to provide feedback during testing phases.

### Data Privacy Compliance

* Ensure compliance with data protection laws (e.g., GDPR, CCPA).
* Provide options for players to opt-in or opt-out of data collection.

## Planning for Scalability and Future Expansions

### Scalability Testing

* Simulate high-load scenarios to test system performance.
* Optimize systems to handle increasing numbers of entities.

### Cross-Platform Considerations

* Keep platform-specific code isolated.
* Test builds on different platforms early to identify issues.

### Modding Support

* Design systems with modding in mind if desired.
* Expose APIs or scripting interfaces for modders.

### Long-Term Maintenance Strategy

* Plan for regular updates and maintenance tasks.
* Allocate time for technology upgrades (e.g., new Unity versions).

## Legal and Licensing

### Asset Licensing Compliance

* Verify licenses for third-party assets, tools, and libraries.
* Keep records of licenses and attributions as required.

### Open-Source Components

* If using open-source software, comply with their licenses (e.g., MIT, GPL).
* Consider the implications of using copyleft licenses in your project.

### Trademarks and Branding

* Ensure your game’s name and branding do not infringe on existing trademarks.
* Register trademarks if necessary.

## Team Collaboration Tools

### Communication Platforms

* Set up team communication channels (e.g., Slack, Discord, Microsoft Teams).
* Establish protocols for meetings, updates, and announcements.

### Project Management Software

* Use tools like Trello, Asana, or Jira to track tasks and progress.
* Break down tasks into manageable units with clear deadlines.

### Asset Management

* Implement version control for large assets if necessary (e.g., Git LFS, Perforce).
* Maintain an organized asset library accessible to team members.

# Expansion Roadmap (Expansion->Milestones->Feature Groups)

## Text to Graphics

### Milestone 1: Text Based Game

#### Feature Group 1: Digital Novel Introduction

#### Feature Group 2: Interactive Fiction with a Test

#### Feature Group 3: The Player Brain & Genetics

#### Feature Group 4: Toddler Text Adventure

### Milestone 2: ASCII Art Foundations

### Milestone 3: Introduction of Interfaces & UI

### Milestone 4: Raster Graphics

## 

### Milestone 1:

### Milestone 2:

### Milestone 3:

### Milestone 4:

## 

### Milestone 1:

### Milestone 2:

### Milestone 3:

### Milestone 4:

## 

### Milestone 1:

### Milestone 2:

### Milestone 3:

### Milestone 4:

## 

### Milestone 1:

### Milestone 2:

### Milestone 3:

### Milestone 4:

## 

### Milestone 1:

### Milestone 2:

### Milestone 3:

### Milestone 4:

## 

### Milestone 1:

### Milestone 2:

### Milestone 3:

### Milestone 4:

## 

### Milestone 1:

### Milestone 2:

### Milestone 3:

### Milestone 4:

## 

### Milestone 1:

### Milestone 2:

### Milestone 3:

### Milestone 4:

## 

### Milestone 1:

### Milestone 2:

### Milestone 3:

### Milestone 4:

## End

# Universe Bible

## World Building

### Space

#### Planetary Orbit

#### Local Moon

#### Solar System

#### Galaxy

#### Universe

### Terrain

#### Landforms

#### Water Bodies

#### Subterranean Features

#### Geologic Forces

#### Natural Resource Distribution

#### Planetary Features

### Environment

#### Climate Patterns

#### Weather Events

#### Natural Disasters

#### Biomes

### Ecosystem

#### Flora

#### Fauna

#### Food Chains

### Geography

#### Countries

#### Regions & Boundaries

#### Significant Locations

#### Travel Routes

## History & Timeline

### Geologic History

#### Formation Periods

#### Natural Disasters Impact

#### Continental Drift and Plate Tectonics

#### Significant Climatic Shifts

### Eras & Ages

#### Prehistoric Age

#### Classical Age

#### Industrial Age

#### Technological Age

#### Future/Possible Ages

### Historical Events

#### Key Wars

#### Political Revolutions

#### Major Inventions or Discoveries

#### Societal Reformations

### Chronological Timeline

#### Important Dates

#### Key Turning Points

#### Event Sequences

### Technological Progression

#### Primitive Technologies

#### Agricultural Developments

#### Industrial Advances

#### Modern Innovations

### Cultural Movements

#### Art and Literature Movements

#### Religious Reforms

#### Social Activism Periods

#### Scientific Enlightenments

## Groups & Organizations

### Races & Cultures

#### Major Ethnic Groups

#### Cultural Characteristics

#### Societal Hierarchies

#### Traditional Roles

### States & Governments

#### Government Types

#### Notable Leaders

#### Policies and Laws

#### Diplomatic Relations

### Businesses & Companies

#### Leading Corporations

#### Economic Influence

#### Industrial Sectors

#### Trade Networks

### Cultural & Religious Institutions

#### Religious Orders

#### Educational Establishments

#### Philosophical Schools

#### Secret Societies

### Military Organizations

#### Army Divisions

#### Intelligence Agencies

#### Defense Alliances

#### Military Hierarchies

## Characters

### Major Characters

#### Protagonists

#### Antagonists

#### Key Advisors

#### Influencers

### Supporting Characters

#### Sidekicks

#### Mentors

#### Minor Villains

#### Civic Leaders

### Background Characters

#### Townsfolk

#### Traders

#### Soldiers

#### Witnesses

### Families & Lineages

#### Noble Houses

#### Famous Bloodlines

#### Lineage Conflicts

#### Cultural Legacies

### Relationships

#### Friendships

#### Rivalries

#### Alliances

#### Romantic Ties

### Roles & Personalities

#### Heroic Archetypes

#### Villainous Archetypes

#### Support Roles

#### Leadership Roles

#### Everyday Roles

### Animals

#### Domesticated Animals

#### Wildlife Species

#### Mythical Creatures

#### Endangered or Rare Species

#### Predators and Prey

## Science & Technology

### Technological Era

#### Innovation Periods

#### Key Milestones

#### Technological Gaps

#### Societal Adoption Rates

### Energy & Power Systems

#### Renewable Sources

#### Industrial Power

#### Unique Energy Resources

#### Energy Distribution

### Transportation & Vehicles

#### Land Vehicles

#### Watercraft

#### Air and Spacecraft

#### Transport Infrastructure

### Communication & IT

#### Communication Networks

#### Information Storage

#### Data Security

#### Advanced Computation

### Weapons, Armor, and Warfare Technology

#### Melee Weapons

#### Ranged Weaponry

#### Defensive Gear

#### War Machines

### Medical and Biotechnology

#### Medical Practices

#### Biotechnological Advances

#### Diseases and Epidemics

#### Pharmaceuticals and Remedies

#### Medical Institutions and Ethics

### Consumer Goods

#### Household Items

#### Personal Items and Fashion

#### Luxury Goods and Status Symbols

#### Recreational Products

#### Food and Beverage Products

## Culture & Economics

### Social Structure & Class Systems

#### Nobility and Royalty

#### Working Classes

#### Social Mobility

#### Cultural Stratification

### Daily Life, Customs, and Traditions

#### Food and Cuisine

#### Clothing and Attire

#### Festivals and Celebrations

#### Daily Routines

### Ideology, Philosophy, and Worldview

#### Core Beliefs

#### Ethical Codes

#### Worldview Perspectives

#### Moral Debates

### Arts, Entertainment, and Cultural Expression

#### Visual Arts

#### Performing Arts

#### Popular Entertainment

#### Cultural Artifacts

### Economic Systems and Trade

#### Currency and Banking

#### Trade Agreements

#### Marketplaces

#### Guilds and Associations

# Art Bible

# Live Operations Plan

# End