4.1 Plan Introduction

Eve is a video game to be developed in Unity, using a pixel art style and a 2.5D rendering system. We are creating this game because we all have an interest in pursuing game development and would like to continue to work on our skills for the future. We will develop this game in Unity, and the pixel art will be done in Promotion NG, with more detailed art done in Procreate and music created in Ableton. We plan to have the Software Development Plan Document complete by Week 7, Software Design Description Document by Week 10, etc. (*fill in with actual milestones)

4.1.1 Project Deliverables

- Software Development Plan Document (Week 07)
- Software Design Description Document (Week 10-12)
- Critical Design Review (Week 12-14)
- User's Manual Final Updates(Week 12)
- Final Product Delivery (Week 16)
- (*fill in with specific things that need to be done)

4.2 Project Resources

4.2.1 Hardware Resources (Development)

- CPU: X64 architecture with SSE2 instruction set support OR Apple M1 or above (Apple silicon-based processors).
- Graphics API: DX10, DX11, and DX12-capable GPUs OR Metal-capable Intel and AMD GPUs.
- Hard Drive Space: ~5-10GB
- Display: 1920 X 1080
- RAM: 8 GB

4.2.2 Software Resources (Development)

- Operating System: Windows & Mac
- Operating Systems Version: Windows 7 (SP1+), Windows 10 and Windows 11, 64-bit versions only OR Mojave 10.14+ (Intel processors), Big Sur 11.0 (Apple silicon-based processors).
- Compiler/Engine: Unity
- Art Software/Graphics: Promotion NG & Procreate
- Audio Creation: Ableton
- Audio Engine: Fmod

4.2.3 Hardware Resources (Execution)

- CPU: X64 architecture with SSE2 instruction set support OR Apple M1 or above (Apple silicon-based processors).
- Graphics API: DX10, DX11, and DX12-capable GPUs OR Metal-capable Intel and AMD GPUs.
- Hard Drive Space: ~1-5GB

- Display: 1920 X 1080
- RAM: 6-8 GB

4.2.4 Software Resources (Execution)

- Operating System: Windows & Mac
- Operating Systems Version: Windows 7 (SP1+), Windows 10 and Windows 11, 64-bit versions only OR Mojave 10.14+ (Intel processors), Big Sur 11.0 (Apple silicon-based processors).

4.3 Project Organization

- Unity Project Management
    - Oversees the entire game development process, coordinating teams, timelines, resources, and ensuring the project stays on track and within scope.
    - Role: CJ
- Scripting for Story
    - Focuses on writing scripts that drive the narrative elements of the game, handling dialogues, branching storylines, and cutscene triggers to enhance the storytelling experience.
    - Role: Gray
- General Scripting
    - Implements core gameplay features and logic, handling tasks such as player controls, NPC behaviors, and game mechanics using C# and Unity's scripting API.
    - Role: Gray and CJ
- Interactive Art
    - Creates and integrates interactive visual assets, such as animations, particle systems, and effects, that respond dynamically to player inputs and game states.
    - Role Brisa
- Scripting for visual design
    - Writes scripts that control visual elements like shaders, lighting, post-processing, and special effects, enhancing the overall aesthetic and feel of the game.
    - Role: Brisa
- Mechanics
    - Designs and scripts the core game systems and mechanics, such as farming, managing the day/night cycle, movement, and more.
    - Role: Matt
- Interaction UI
    - Develops the user interface (UI) elements, focusing on intuitive design and scripting to handle input, feedback, and interaction between the player and the game systems.
    - Role: Matt
- General game play tools
    - Builds reusable tools and systems for the development team, such as level editors, asset managers, and debugging tools, to streamline the game creation process.
    - Role Matt

4.4 Project Schedule

4.4.1 PERT/GANTT Chart

- Link to Chart
  - https://docs.google.com/spreadsheets/d/1h-3aPss_imEJ1yIpYtgKFFn24p-yDgOpw7Lwemw5ZWo/edit?usp=sharing

4.4.2 Task/Resource Table

- UNLESS SPECIFIED hardware needed will be the same as what is listed in section 5.5.1

TASK	HARDWARE/SOFTWARE REQUIRED
Initial Unity 3D setup	- Software - Unity
Setup Tilemap	- Software - Unity
Create Layers/Tags for Collisions and interaction	- Software - Unity
Camera	- Software - Unity
Player Controller	- Software - Unity
Initial Game Manager	- Software - Unity
Farming Scripts	- Software - Unity
Difficulty Scaling	- Software - Unity
Task Manager	- Software - Unity
Music Manager	- Software - Unity - Fmod
Cut Scenes	- Software - Unity
Initial Pixel Draft	- Software - Promotion NG
Pixel Tilemap Draft	- Software - Promotion NG
Setting up 2.5D effects	- Software - Unity

Sound Effects	- Software - Ableton
Farming Sounds	- Software - Ableton
Music	- Software - Ableton
Main Theme	- Software - Ableton

5.1 Introduction

EVE shall be a video game designed in Unity using a 2.5D rendering system, which Unity natively supports. No outside libraries are intended for use, excluding the ones already supported by Unity and the ones that are readily accessible in its provided list of libraries. In terms of rendering, most gameplay will be in a pixel artstyle, with character closer-up interactions containing prerendered art of the individual. Components in terms of how the game should be organized would be separated into a player character manager, a world manager (for keeping track of time, physics, world interaction, etc.), and a plot manager (which would keep track of any character interactions and plot advancements to direct events in the world).

5.2 CSC Component Breakdown

5.2.1 Unity

5.2.1.1 Unity Engine
5.2.1.2 Graphical User Interface
5.2.1.3 Game Manager
5.2.1.4 Game Mechanics
5.2.1.5 Inventory Management
5.2.1.6 Time Management
5.2.1.7 Farming Tools
5.2.1.8 Quests and Objectives

5.2.2 Pro Motion NG

Will be used to create various assets and background images for the game, including most in-game portrayals of player and non-player characters, items, background images, menu and GUI elements, and interaction animations.

5.2.3 Procreate

Will be used to create more detailed art depictions of the various player characters and non-player characters in a traditional art style, as well as interaction UI and concept art

5.2.4 Ableton

Will be used to create various background music tracks, which will be played constantly throughout the game and will dynamically change based on situations detailed by the Unity engine

5.2.5 Fmod

Will be used to generate various in-game audio cues and sounds to give players feedback based upon their actions, or telegraph certain events based on situations detailed by the Unity engine

5.3 Functional Requirements by CSC

5.3.1 Unity

5.3.1.1 Unity- Engine

5.3.1.1 The Unity subsystem shall be used as the primary game engine for rendering environments.
5.3.1.2 The Unity subsystem shall handle physics-based interactions such as collisions, gravity, and movement.
5.3.1.3 The Unity subsystem shall provide support for scripting in C# to control player interactions, game mechanics, and UI elements.
5.3.1.4 The Unity subsystem will include built-in performance profiling tools to monitor framerate, memory usage, and processing overhead.
5.3.1.5 The Unity subsystem shall integrate with third-party libraries and plugins, including Fmod for audio and Pro Motion NG for art assets.
5.3.1.6 The Unity subsystem shall manage lighting and shadow rendering for both daytime and nighttime gameplay environments.
5.3.1.7 The Unity subsystem will support particle systems for rendering weather effects like rain, snow, and wind.
5.3.1.8 The Unity subsystem shall allow the import and management of pixel art assets created in Pro Motion NG and Procreate.

5.3.1.2 Graphical User Interface

The GUI is intended to be the forefront of the project, given that graphical feedback of a game system is imperative. This subsystem will manage the windows for a start menu, pause menu, quit and play buttons, responsive input requirements, and an options menu

5.3.1.2.1 The GUI subsystem shall display a window for the main application

5.3.1.2.2 The GUI subsystem shall display a window to provide details of operation (a "help" window)

5.3.1.2.3 The GUI subsystem shall display a window which will allow users to change input settings or make graphical adjustments (an "options" window)

5.3.1.2.4 The GUI subsystem shall include a button which will quit the application

5.3.1.2.5 The GUI subsystem shall include a menu bar that will pause all physics and player interaction when pressed, and display the aforementioned windows in 5.3.1.2 and 5.3.1.3, and the button in 5.3.1.4 (a "pause" menu)

5.3.1.2.6 The GUI subsystem shall react to mouse clicks on displayed buttons

5.3.1.2.7 The GUI subsystem shall react to keyboard inputs in conjunction with moving displayed game components

5.3.1.2.8 The GUI subsystem shall contain a button to start the main application

5.3.1.2.9 The GUI subsystem shall contain a window which will display the menu bar mentioned in 5.3.1.2.10, as well as the button in 5.3.1.1.8 (a "start menu")

5.3.1.2.11 The GUI subsystem shall contain a sprite which will be rendered in reaction to the buttons pressed as in 5.3.1.7 ("player movement")

5.3.1.2.12 The GUI subsystem shall display a menu which will display various sprites which are contained in the player's inventory

5.3.1.3 Game Manager

5.3.1.3.1 The GM subsystem shall contain references to all keyboard and mouse inputs in relation to how they would influence the GUI

5.3.1.4 Game Mechanics

5.3.1.4.1 The game mechanics subsystem shall support planting various crops.

5.3.1.4.2 The game mechanics subsystem shall include a crop growth cycle based on in-game time, where crops progress through different growth stages.

5.3.1.4.3 The game mechanics subsystem will allow crops to wither if not harvested within a specified

period.
5.3.1.4.4 The game mechanics subsystem shall support crop harvesting when they reach maturity.
5.3.1.4.5 The game mechanics subsystem shall support tilling the soil before planting crops.
5.3.1.4.6 The game mechanics subsystem shall include weather conditions, affecting crops and player actions (e.g., rainwater crops automatically).
5.3.1.4.7 The game mechanics subsystem shall support selling crops and animal products for in-game currency.

5.3.1.5 Inventory Management

5.3.1.5.1 The inventory subsystem shall allow the player to store seeds, crops, tools, and other items.
5.3.1.5.2 The inventory subsystem shall limit the number of items the player can carry at one time based on defined slots.
5.3.1.5.3 The inventory subsystem shall support the transfer of items between the player and storage containers such as chests or barns.
5.3.1.5.4 The inventory subsystem shall display the current item selection for use by the player.
5.3.1.5.5 The inventory subsystem shall allow players to stack identical items to save space.
5.3.1.5.6 The inventory subsystem shall categorize items for easy access (e.g., crops, seeds, tools).
5.3.1.5.7 The inventory subsystem shall allow for quick access to tools via a toolbar or hotkey system.
5.3.1.5.8 The inventory subsystem shall support a system for crafting new items using resources stored in the inventory.

5.3.1.6 Time Management

5.3.1.6.1 The time management subsystem shall simulate in-game time, progressing through day and night cycles.

5.3.1.6.2 The time management subsystem shall include seasons that change based on the passage of in-game days.

5.3.1.6.3 The time management subsystem shall include a calendar system where players can track upcoming events (e.g., festivals).

5.3.1.6.4 The time management subsystem shall include an alarm system where players can be notified of significant in-game events.

5.3.1.6.5 The time management subsystem shall include a system where in-game tasks (e.g., watering crops) must be completed within specific time windows.

5.3.1.7 Farming Tools

5.3.1.7.1 The tool subsystem shall allow the player to equip and use farming tools such as a hoe, watering can, scythe, and axe.
5.3.1.7.2 The tool subsystem shall allow tools to be upgraded using in-game materials.
5.3.1.7.3 The tool subsystem shall allow the use of tools to affect the environment (e.g., cutting trees, breaking rocks).
5.3.1.7.4 The tool subsystem shall include a repair mechanic for tools after they degrade with use.
5.3.1.7.5 The tool subsystem shall include a system for crafting new tools using resources collected by the player.
5.3.1.7.6 The tool subsystem will include a fishing tool, allowing the player to catch fish from bodies of water.
5.3.1.7.7 The tool subsystem shall include a mechanic for fertilizing crops to improve yield.

5.3.1.8 Quests and Objectives

5.3.1.8.1 The quest subsystem shall include a main storyline quest where players work to meet a quota provided by their employer.

5.3.1.8.2 The quest subsystem shall include side quests.

5.3.1.8.3 The quest subsystem shall reward players with in-game currency or special items for completing quests.

5.3.1.8.4 The quest subsystem shall include repeatable daily tasks such as watering crops and collecting resources.

5.3.1.8.5 The quest subsystem shall allow players to track their current objectives via an in-game journal.

5.3.2 Pro Motion NG - Pixel Art Software

5.3.2.1 The Pro Motion NG subsystem will be used for creating pixel art assets, including sprites, tiles, and UI elements.
5.3.2.2 The Pro Motion NG subsystem shall support exporting pixel art in compatible formats for Unity import.

5.3.2.3 The Pro Motion NG subsystem shall support creating multi-layered animations for player characters and NPCs.
5.3.2.4 The Pro Motion NG subsystem shall allow batch editing of pixel art assets to ensure consistency across game assets.
5.3.2.5 The Pro Motion NG subsystem shall support color palette management, allowing the creation of variations of the same sprite assets.

5.3.3 Procreate - Art Software

5.3.3.1 The Procreate subsystem will be used for creating concept art, illustrations, and hand-drawn textures.
5.3.3.2 The Procreate subsystem shall support exporting high-resolution textures for use in Unity.
5.3.3.3 The Procreate subsystem shall be used to create character portraits, used in player-NPC dialogue and interaction.
5.3.3.4 The Procreate subsystem shall allow the creation of UI art elements such as buttons, inventory icons, and menu backgrounds.
5.3.3.5 The Procreate subsystem shall support layer management and transparency for easy integration with Unity's sprite system.

5.3.4 Ableton - Music Software

5.3.4.1 The Ableton subsystem will be used for composing and producing background music for the game.
5.3.4.2 The Ableton subsystem will allow the creation of a dynamic soundtrack, adapting to different game conditions such as seasons, weather, or events (e.g., festival music, rainy day music).
5.3.4.3 The Ableton subsystem shall support the creation of looping ambient tracks for various farm locations (e.g., fields, barns, town).
5.3.4.4 The Ableton subsystem will support MIDI integration for composing adaptive music layers that can transition smoothly between game states.
5.3.4.5 The Ableton subsystem will allow the export of music tracks in formats compatible with the Fmod audio engine for integration into the Unity engine.
5.3.4.6 The Ableton subsystem shall include sound effects creation, allowing for the design of distinct in-game sounds (e.g., harvesting, weather).
5.3.4.7 The Ableton subsystem shall allow collaboration between sound designers through the use of project files that can be shared and edited.

5.3.5 Fmod - Audio Engine

5.3.5.1 The Fmod subsystem will be used for managing all in-game audio, including music, sound effects, and ambient sounds.
5.3.5.2 The Fmod subsystem shall support dynamic audio mixing, adjusting volume levels based on game events or player actions.
5.3.5.3 The Fmod subsystem shall handle the implementation of adaptive music created in Ableton, ensuring smooth transitions between different music states.
5.3.5.4 The Fmod subsystem will support spatial audio for 3D sound effects, enhancing immersion by simulating the distance and direction of sounds.
5.3.5.5 The Fmod subsystem shall manage a library of sound effects triggered by player actions (e.g., using tools, and harvesting crops).
5.3.5.6 The Fmod subsystem shall include a real-time parameter control system, allowing game state changes (e.g., time of day, weather) to influence the soundscape.
5.3.5.7 The Fmod subsystem shall allow for the integration of environmental audio effects, such as wind, rain, and wildlife, which change dynamically with player location.
5.3.5.8 The Fmod subsystem will support multi-channel output for surround sound setups, enhancing the player's auditory experience.

5.4 Performance Requirements

5.4.1 Cross-Platform Integration

The game will be playable on both 32-bit and 64-bit Windows and Mac computers.

5.4.2 Preloaded Scenes

This game will load scenes before gameplay to reduce lag.

5.4.3 Light Weight

This game will be limited to 50GB to reduce the storage space required to play.

5.4.4 Minimal Loading Times

This game shall not have an unmoving loading screen longer than 10 seconds.

5.4.5 Multiple Saves

This game will have the capacity to hold multiple saves from players without compromising the integrity of the performance quality.

5.4.6 High Visual Quality

This game will be optimized visually for differing screen sizes and refresh rates.

5.5 Project Environment Requirements

5.5.1 Development Environment Requirement

Following are the hardware requirements for Eve:

- CPU: X64 architecture with SSE2 instruction set support OR Apple M1 or above (Apple silicon-based processors).
- Graphics API: DX10, DX11, and DX12-capable GPUs OR Metal-capable Intel and AMD GPUs.
- Hard Drive Space: ~5-10GB
- Display: 1920 X 1080
- RAM: 8 GB
-

Following are the software requirements for Eve:

- Operating System: Windows & Mac
- Operating Systems Version: Windows 7 (SP1+), Windows 10 and Windows 11, 64-bit versions only OR Mojave 10.14+ (Intel processors), Big Sur 11.0 (Apple silicon-based processors).
- Compiler/Engine: Unity
- Art Software/Graphics: Promotion NG & Procreate
- Audio Creation: Ableton
- Audio Engine: Fmod

5.5.2 Execution Environment Requirement

Following are the hardware requirements for Eve:

- CPU: X64 architecture with SSE2 instruction set support OR Apple M1 or above (Apple silicon-based processors).
- Graphics API: DX10, DX11, and DX12-capable GPUs OR Metal-capable Intel and AMD GPUs.
- Hard Drive Space: ~1-5GB

- Display: 1920 X 1080
- RAM: 6-8 GB

Following are the software requirements for Eve:

- Operating System: Windows & Mac
- Operating Systems Version: Windows 7 (SP1+), Windows 10 and Windows 11, 64-bit versions only OR Mojave 10.14+ (Intel processors), Big Sur 11.0 (Apple silicon-based processors).