Department of Computer Science & Engineering
The University of Texas at Arlington

System Requirements Specification
CSE 4316: Senior Design I
Summer 2022

Euporie Studios
Project Calamity

Matthew Green
Robert Kembel
Jacob Lanham
Bradley Miettinen

Revision History

Revision	Date	Author(s)	Description
0.1	07.14.2022	JL	document creation
0.2	07.29.2022	JL, MG, RK, BM	customer requirements initial draft
1.0	07.31.2022	JL, MG, RK, BM	finalized version
1.1	12.9.2022	JL	Updated to fit changes in game

Contents

List of Figures

1 Product Concept

This section provides a high-level statement of Project Calamity concept - what it is intended to do and how it is intended to be used.

1.1 Purpose and Use

The purpose of this project is to contribute an original piece of art to the entertainment industry. We wish to pull the audience into our imagination where they can engage with unique experiences designed to challenge, inspire, and entertain. The result will be a interactive expression of our creativity which will test and grow the development skills of the team.

Our tool of choice for this contribution is a sci-fi horror first-person-shooter video game. It is intended to be played on a PC - preferably in a dark room!

1.2 Intended Audience

The intended audience is people who enjoy horror games and own a PC running a Windows 10 or newer.

2 Product Description

2.1 Features & Functions

This section provides the reader with an overview of Project Calamity. The basic components which make up the game are explained here.

Project Calamity is a PC video game which takes input from the player via mouse and keyboard. Major system components for the game are represented in Figure 1 below.

The first three components of the diagram represent the input system for the game. The Player interacts with the Controller (mouse and keyboard), which in turn interacts with the game software to control the Player Character. Next there are a series of game components the player will interact with which make up the bulk of the functionality of the game.

The Environment includes all non-character assets which make up the world. Within the environment, the player will find enemies they need to fight in order to survive.

As the player explores the world, they will require a means of gathering resources - such as equipment - to face the challenges ahead. This will be done via a Player Management System. At fixed points in the game, the player can access terminals to equip and craft gear, or some other kind of item. Additionally, a quest system will award the player with various materials used for crafting. The quest system tracks which quests are unfinished, in progress, or complete.

The Inventory System will be implemented to allow both players and NPC's to carry and access a fixed number of items. The player can use this to equip and remove clothes, weapons, money, etc. as well as trade items with NPC's.

The Combat System controls how players and enemies fight within the game as well as how the environment and characters respond to damage.

Finally, the User Interface is what the Player sees on screen as the Player Character interacts which each of the various game components. This includes everything from menus and dialogue boxes to health bars and navigation aides.

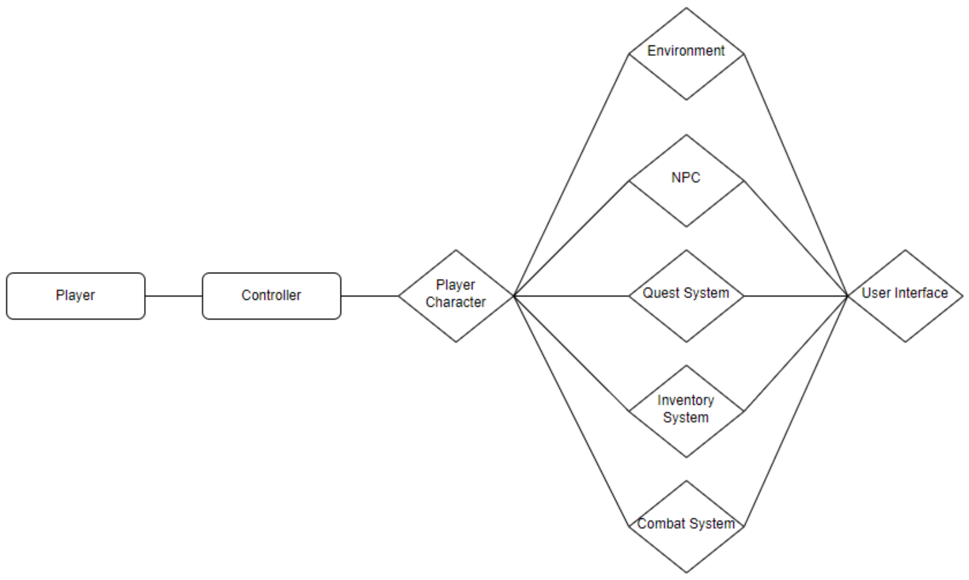

Figure 1: System Diagram

2.2 External Inputs & Outputs

Inputs will be keyboard and mouse. Outputs will be using a standard 2D raster monitor for graphics and using speakers, headphones, or any other sound devices for in-game sounds.

2.3 Product Interfaces

This is the current look and implementation for the inventory menu. Currently the overall height of the vest and backpack inventory space is not being taken into account by the VerticalLayoutGroup component on the parent GameObject of those UI elements.

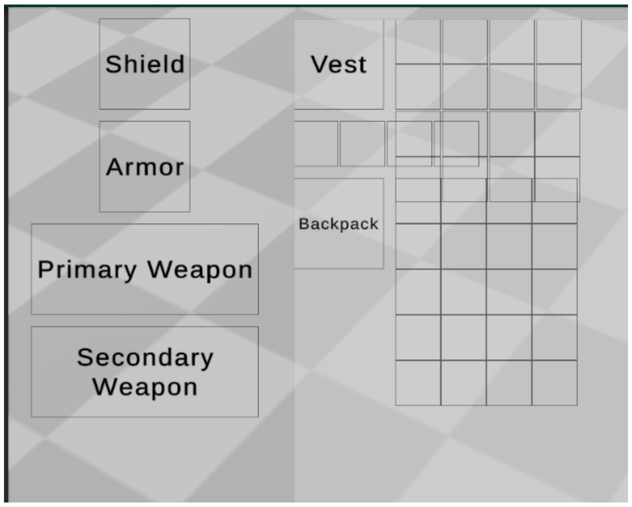

Figure 2: Project Calamity: Prototype inventory menu.

This is what the menu should like after fixing the overall height not being taken into account.

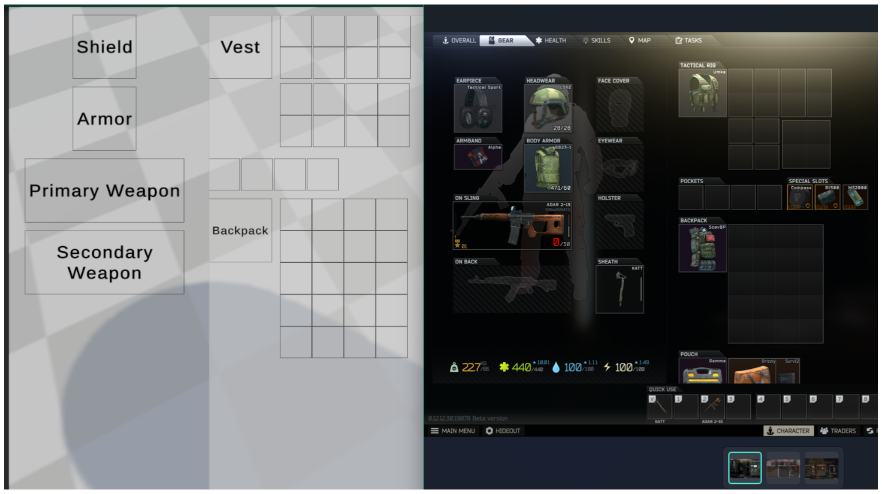

Figure 3: Project Calamity: Prototype inventory menu after readjusting UI element values during runtime.

3 Customer Requirements

Project Calamity is a horror-themed first person shooter video game. The purpose of the game is to entertain in the context of a unique story and gameplay loop. The following requirements follow from this goal and are what the player can expect of their experience in-game.

3.1 Sci-fi Horror Environment Design

3.1.1 Description

The foundation of the player experience is an environment that is both interesting and induces fear. The player can expect such design choices to influence the look and feel of their world which is also embedded in a sci-fi aesthetic.

3.1.2 Source

Robert Kembel (team member).

3.1.3 Constraints

The lighting conditions should be adjusted to control player visibility as needed to fit the atmosphere of the game at a specific point in the story. For example a jump scare can be induced by lowering visibility of areas where enemies are likely to attack.

The environment design should created with the purpose of increasing the anxiety and anticipation of the player as needed for the story. For instance, while hunting a monster, the player could find remnants the monsters previous attacks in the environment.

The assets used to construct the world must follow or be complementary to a science fiction aesthetic.

3.1.4 Standards

None

3.1.5 Priority

Critical

3.2 Horror-Themed Enemy Character Design

3.2.1 Description

The game will contain enemy characters and the design of the enemies will contribute to the sci-fi horror atmosphere of the game.

3.2.2 Source

Robert Kembel (team member)

3.2.3 Constraints

The design of enemy characters should be compatible with the needs of the story and the sci-fi horror aesthetic.

3.2.4 Standards

None

3.2.5 Priority

High

3.3 Story

3.3.1 Description

The various events of the game will be tied together with an original story to guide the player through the environment as well as provide context to events, quests, and other game elements.

3.3.2 Source

Matthew Green (team member), Robert Kembel (team member).

3.3.3 Constraints

The game must include storytelling elements throughout to provide context. Storytelling elements can include but are not limited to text screens, cut-scenes, dialogue and environmental artifacts.

3.3.4 Standards

None

3.3.5 Priority

High

3.4 Player Controller

3.4.1 Description

The system that implements all the ways that the user can control the player character, open menus, and generally interact with the game's world.

3.4.2 Source

Matthew Green (team member).

3.4.3 Constraints

Ability to walk in all 4 directions; forward, backward, right, and left as well as the ability to combine directions. Ability to manipulate the camera's rotation on 2 axis; pitch and yaw. Ability to go crouched and prone. Accompanying animations and/or sound for each action the player performs. Ability t open menus. Ability to shoot, reload, and aim down sights on current weapon. Ability to switch between equipped weapons. Ability to melee hit using fists or current weapon.

3.4.4 Standards

None

3.4.5 Priority

Critical

3.5 Inventory

3.5.1 Description

The player will have several different equipment slots to hold weapons, armor, and items. They will also be able to see how many credits they have and drop and organize items within their inventory. Ammunition, magazines, and batteries must be inside the vest or pockets in order to reload. Any of these items inside backpacks will not be used when reloading.

3.5.2 Source

Matthew Green (team member).

3.5.3 Constraints

A primary and secondary firearm slot. Shield Slot. Armor Slot. Vest Slot. Backpack Slot. Pockets for persistent storage. As well as a credit count somewhere. Tetris style inventory space for the equipped vest and backpack as well as pockets; what you have to do manage the positioning, rotation, and size of each item.

Inspiration

The different sources of inspiration from where the functionality of the inventory was decided from.

EscapeFromTarkov's player inventory and stash allows you and greatly requires you to heavily manage by using the smallest container slots possible and rotating items for more efficient storage. You are able to get more inventory space by equipping different tactical rigs and backpacks, in their respective slots.

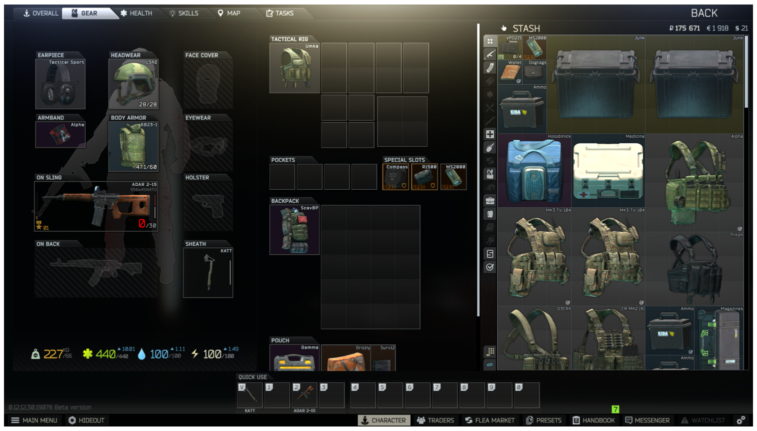

Figure 4: EscapeFromTarkov: Player inventory and stash.

Resident Evil's player inventory is much like EscapeFromTarkov, except you just have a single large rectangular container that can be upgraded.

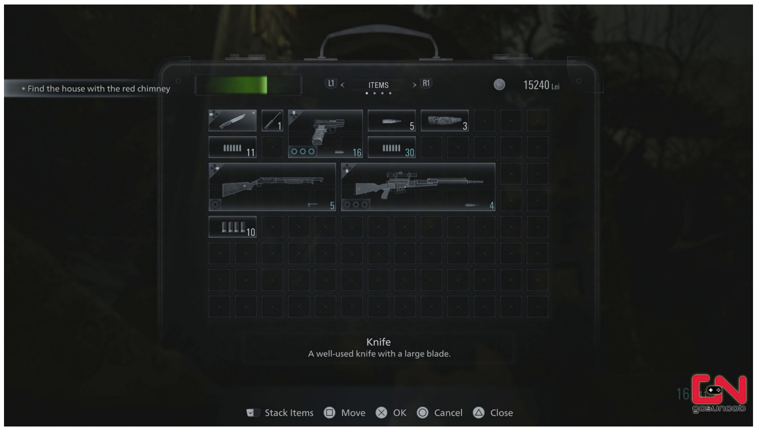
Find the house with the red chimney
L1
ITEMS
R1
15240 Lei
Knife
A well-used knife with a large blade.
Stack Items
Move
OK
Cancel
Close

Figure 5: Resident Evil 8: Village: Player inventory.

3.5.4 Standards

None

3.5.5 Priority

Critical

3.6 Player Management Stations

3.6.1 Description

A station that can accessed by the player in-game, allowing them to perform a large number of functions such as accessing the quest system, organized by different modules.

3.6.2 Source

Matthew Green (team member).

3.6.3 Constraints

There is different modules that are capable of different functionalities. There is two types of stations, the type determines how many modules are accessible. Module functionality can also be limited by the type of station.

Differences Between Types

There are two types α and β stations. All α stations have access to all modules and can allow the player to instantly transport to another pre-discovered α station. The α stations also are not placed in areas where the player can get into combat. While β are only found in combat zones, most likely in loot rooms, to perform upgrades and crafting in the field, using resources only in the player's inventory; they do not have access to α station's stash. So the player can only use brought-in or found in-field resources, with the exception of credits. Both types of stations need to be activate to use and α stations must be activated to be transported to. All stations have certain modules enabled, such as the Memory Module, allowing the player to save and load the game.

There is an α station inside your crashed spaceship. However, you start with no other α stations activated; so you have to explore around.

The α stations, when accessed, also heal the player and respawn all enemies except for special enemies and bosses.

Inspiration

The different sources of inspiration from where the functionality of a station was decided from.

Dead Space's player upgrade stations; where you can upgrade your weapons and player's max health.

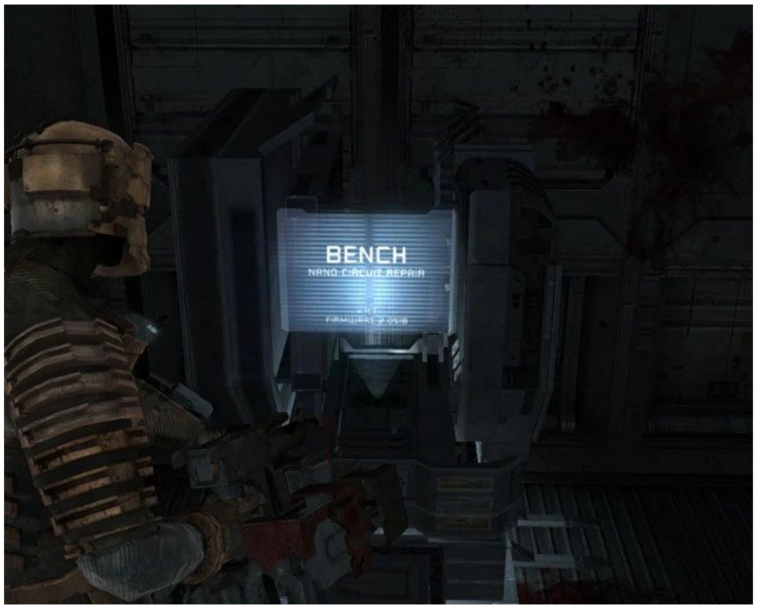

Figure 6: Dead Space: Player upgrade station or "bench".

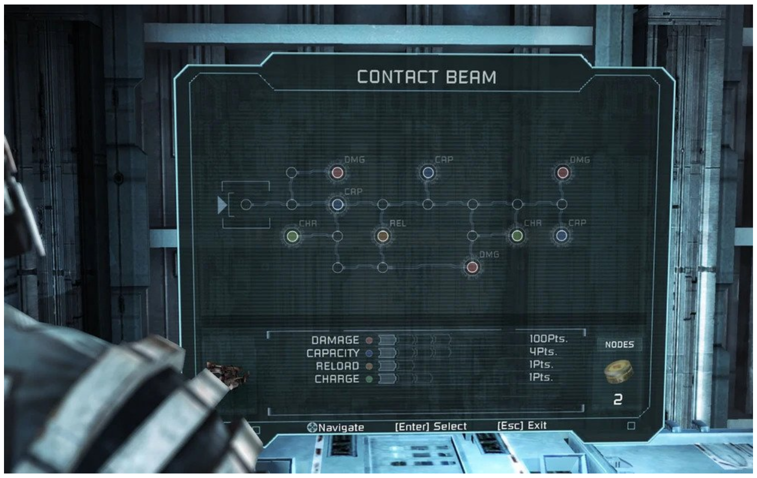

Figure 7: Dead Space: Player upgrade station or "bench" weapon upgrade menu.

Dark Souls' bonfires; which heal the player, includes a stash, allows the player to increase max health, and respawns all enemies.

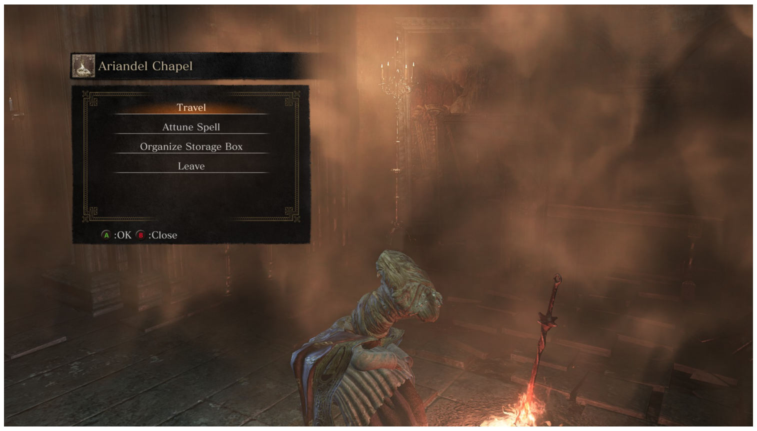

Figure 8: Dark Souls III: Generic bonfire, akin to β stations.

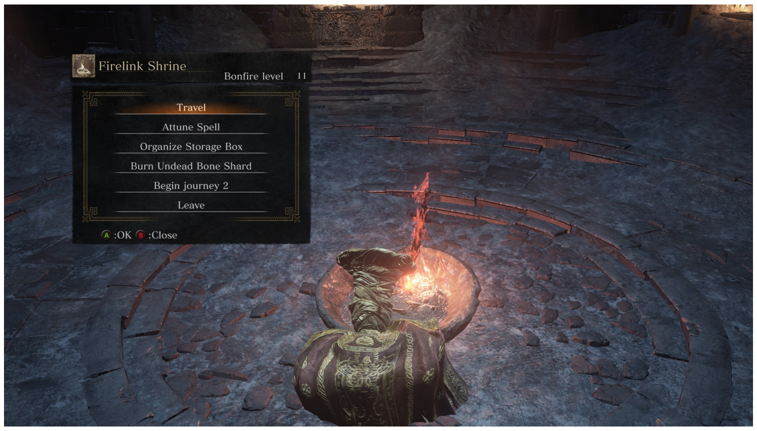

Figure 9: Dark Souls III: Firelink bonfire, akin to α stations, has increased capabilities from generic bonfires.

Subnautica's fabricators, which are basically 3D printers that can print in any material provided; they allow you turn raw resources into other items, and sometimes the items can be used again in another recipe, into different items.

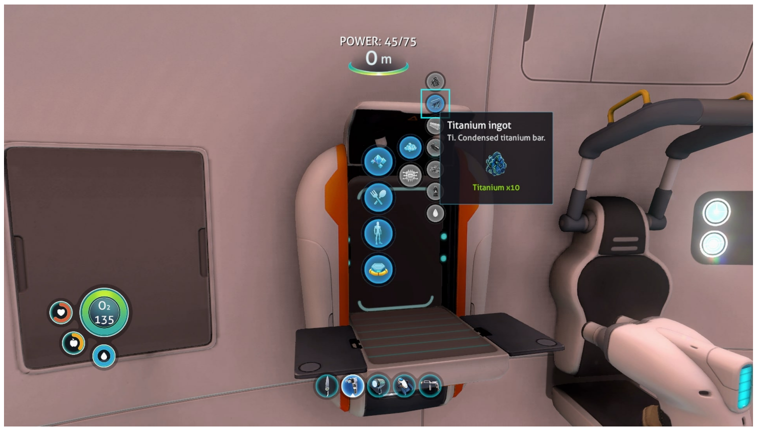

Figure 10: Subnautica: Crafting w/ a Fabricator.

Modules

α - means that this module can be found at an α PMS. β - means that this module can be found at a β PMS.

- Suit Module *($\alpha + \beta$)*
  - Permanent health upgrade, using credits and/or found resources
  - Movement speed upgrade, using credits and/or found resources
- Weapon Upgrade Module *($\alpha + \beta$)*
  - Upgrade equipped weapons, using credits or found resources
- Stash Module *(α)*
  - Upgrade stash size, using credits and/or found resources
  - Store resources
  - Store purchased weapons
- Transportation Module *(α)*
  - Travel between discovered α stations
- Mission Module *($\alpha + \beta$)*
  - Get missions from here, from other people on the planet, asking for your help
  - Turn in mission items
  - Collect mission rewards
- Market Module *(α)*
  - Use credits to purchase either single-use or multi-use files to print items. A weapon would be single-use, while a magazine for it would be a multi-use file.

  - Displays the resources needed to craft item, and compares resources on your character and your total resources including the stash.

- Crafting Module *($\alpha + \beta$)*
  - Allows the player to used purchased files, to print items, using found resources in inventory. Can also use resources found inside your stash if at an α station.
  - Example: Subnautica, you essentially use a sci-fi 3D printer, to turn raw resources into usable items. In our case the credits act as a fee to get access to the "file" of the item you want to print. The resources are the raw resources needed to print the item and are consumed upon crafting.
  - Recycler, break down found weapons and items, into 50% of there original resources and items used in they're printing.

- Memory Module *($\alpha + \beta$)*
  - Save the game.
  - Load another save.
  - Exit the game.

3.6.4 Standards

None

3.6.5 Priority

Critical

3.7 Enemy Controllers

3.7.1 Description

The different AI systems that control the different enemies found in-game.

3.7.2 Source

Matthew Green (team member).

3.7.3 Constraints

The enemy AIs will be able to guard, patrol, investigate, and attack. Guarding is just the enemy sitting and looking out for threats. Patroling is the same as guarding except they are on a set route that walk along. Investigating is performed when an enemy hears or sees something out of the ordinary. Attacking can be broken down further based on their intelligence and way of attacking. If the enemy is intelligent they will try and get cover and use group tactics to attack the player. The more unintelligent enemy will not use cover and not work as a group. If the enemy's attack is ranged then it will shoot from a distance. If it's attack is melee-based then it will close the distance between the player in order to be able to attack. The controllers will also decide when to play what animations corresponding to the actions the AI is performing.

3.7.4 Standards

None

3.7.5 Priority

Critical

3.8 UI

3.8.1 Description

Visual interface components for the player to both interact with and to display information such as health status. The UI will provide a means of observing other non-environmental game components such as the inventory, quest system, combat system etc.

3.8.2 Source

Bradley Miettinen (team member).

3.8.3 Constraints

Crucial information relevant to the the games components must be properly displayed. This includes but is not limited to information like player health and munitions.

3.8.4 Standards

None

3.8.5 Priority

Critical

3.9 Individual Quest Items

3.9.1 Description

Special items that keep track of progress of various tasks. Completion of quests will award the player with additional resources and possibly experience if a level system is added.

3.9.2 Source

Jacob Lanham (team member).

3.9.3 Constraints

None

3.9.4 Standards

None

3.9.5 Priority

Critical

3.10 Quest Management System

3.10.1 Description

A special inventory system that gives the player the ability to hold multiple quests, complete them in an arbitrary manner, and view the details of currently held quests.

3.10.2 Source

Jacob Lanham (team member).

3.10.3 Constraints

The player must have a way to accept, deny, view progress of, cancel, and complete multiple quests simultaneously within the user interface.

3.10.4 Standards

None

3.10.5 Priority

High

3.11 Quest Reward System

3.11.1 Description

When quests are marked as complete, resources, experience, and any other items for completing the quest are given to the player.

3.11.2 Source

Jacob Lanham (team member).

3.11.3 Constraints

The player must have room in their inventory for reward items in order to collect them, and there must be a place near the player to drop the quest rewards when completed.

3.11.4 Standards

None

3.11.5 Priority

High

3.12 Player Level System

3.12.1 Description

This is a back end system which controls the level of the Player Character and adjusts the game difficulty accordingly.

3.12.2 Source

Team Discussion

3.12.3 Constraints

There needs to be a defined scope for the value of the player's level and the game difficulty.

3.12.4 Standards

None

3.12.5 Priority

Low

4 Packaging Requirements

Product will ship as an executable file from one or multiple sources.

4.1 Github Releases

4.1.1 Description

Product prototype can be acquired via a downloadable executable file from our *GitHub releases page*.

4.1.2 Source

Team Discussion

4.1.3 Constraints

Available for Windows 10 or higher.

4.1.4 Standards

None

4.1.5 Priority

Critical

4.2 Steam Release

4.2.1 Description

Product demo and final release can be acquired via the steam marketplace.

4.2.2 Source

Team Discussion

4.2.3 Constraints

Available for Windows 10 or higher.

4.2.4 Standards

None

4.2.5 Priority

Low

5 Performance Requirements

This section describes all performance requirements for Project Calamity, including minimum FPS and what devices it shall run on.

5.1 Performance On High-End Systems

5.1.1 Description

Game should be able to be run on a high-end system.

5.1.2 Source

Matthew Green (team member).

5.1.3 Constraints

An average of 60 FPS, while in-game.

5.1.4 Standards

None

5.1.5 Priority

High

5.2 Performance On Low-End Systems

5.2.1 Description

Game should be able to be run on a medium to low-end system.

5.2.2 Source

Matthew Green (team member).

5.2.3 Constraints

An average of 30 FPS, while in-game.

5.2.4 Standards

None

5.2.5 Priority

Low

6 Safety Requirements

Project Calamity is a software product and will not need additional safety requirements beyond standard lab procedures.

6.1 Laboratory equipment lockout/tagout (LOTO) procedures

6.1.1 Description

Any fabrication equipment provided used in the development of the project shall be used in accordance with OSHA standard LOTO procedures. Locks and tags are installed on all equipment items that present use hazards, and ONLY the course instructor or designated teaching assistants may remove a lock. All locks will be immediately replaced once the equipment is no longer in use.

6.1.2 Source

CSE Senior Design laboratory policy

6.1.3 Constraints

Equipment usage, due to lock removal policies, will be limited to availability of the course instructor and designed teaching assistants.

6.1.4 Standards

Occupational Safety and Health Standards 1910.147 - The control of hazardous energy (lockout/tagout).

6.1.5 Priority

Critical

6.2 National Electric Code (NEC) wiring compliance

6.2.1 Description

Any electrical wiring must be completed in compliance with all requirements specified in the National Electric Code. This includes wire runs, insulation, grounding, enclosures, over-current protection, and all other specifications.

6.2.2 Source

CSE Senior Design laboratory policy

6.2.3 Constraints

High voltage power sources, as defined in NFPA 70, will be avoided as much as possible in order to minimize potential hazards.

6.2.4 Standards

NFPA 70

6.2.5 Priority

Critical

6.3 RIA robotic manipulator safety standards

6.3.1 Description

Robotic manipulators, if used, will either housed in a compliant lockout cell with all required safety interlocks, or certified as a "collaborative" unit from the manufacturer.

6.3.2 Source

CSE Senior Design laboratory policy

6.3.3 Constraints

Collaborative robotic manipulators will be preferred over non-collaborative units in order to minimize potential hazards. Sourcing and use of any required safety interlock mechanisms will be the responsibility of the engineering team.

6.3.4 Standards

ANSI/RIA R15.06-2012 American National Standard for Industrial Robots and Robot Systems, RIA TR15.606-2016 Collaborative Robots

6.3.5 Priority

Critical

7 Maintenance & Support Requirements

Maintenance of the Product will include addressing any bugs and managing any marketplace considerations.

7.1 Documentation

7.1.1 Description

Documentation with the aim of explaining nomenclature, how the project operates and its overall design.

7.1.2 Source

Brad Miettinen (team member)

7.1.3 Constraints

Possible constraints to be considered are the potential for high man-hours involved both with understanding project architecture. Additionally, omissions in project documentation will be unavailable for clarification by the project team.

7.1.4 Standards

None

7.1.5 Priority

Future

7.2 Bug Fixes

7.2.1 Description

Developers will periodically update the game to fix reported bugs.

7.2.2 Source

Robert Kembel (team member)

7.2.3 Constraints

Bugs will have to be prioritized such that those that have the greatest effect on the user experience are handled first.

7.2.4 Standards

None

7.2.5 Priority

Future

8 Other Requirements

8.1 Coding Standard

8.1.1 Description

All files shall use the coding standards as discussed by the team to allow for easier maintainability and readability, as well as decreasing compile and build times.

8.1.2 Source

Matthew Green (team member).

8.1.3 Constraints

Follow Microsoft's C# conventions. Use proper namespaces corresponding to Assembly Definitions and add new Assembly Definitions and namespaces for increased organization and to increase compilation and building times. Use the *C# New Behaviour Script template* to enforce code organization standards.

8.1.4 Standards

Microsoft's C# identifier naming rules and conventions
Microsoft's C# Naming Guidelines
Microsoft's C# Coding Conventions

8.1.5 Priority

High

9 Future Items

All features and requirements that cannot be completed by the prototype version, such as fixing non-critical bugs and general maintenance of the project.

9.1 Documentation

9.1.1 Description

Documentation with the aim of explaining how the project operates, nomenclature and overall design.

9.1.2 Source

Source

9.1.3 Constraints

Possible constraints to be considered are the potential for high man-hours involved both with understanding project architecture and actual bug fixes. Additionally, omissions in project documentation will be unavailable for clarification by the project team.

9.1.4 Standards

None

9.1.5 Priority

Future

9.2 Bug Fixes

9.2.1 Description

Developers will periodically update the game to fixed reported bugs.

9.2.2 Source

Robert Kembel (team member)

9.2.3 Constraints

Bugs will have to be prioritized such that those that have the greatest effect on the user experience are handled first.

9.2.4 Standards

None

9.2.5 Priority

Future

References