**Blight Brew**

**Technical Design Document**

Version 0.2

**Version History**

|  |  |  |
| --- | --- | --- |
| **Version Number** | **Edited By** | **Date** |
| 0.1 | Alexander Bowerman | 2022/11/10 |
| 0.2 | Alexander Bowerman | 2022/11/11 |
| 0.3 | Perrin Peterson | 2022/12/08 |
| 0.4 | Perrin Peterson | 2023/24/02 |
|  |  |  |
|  |  |  |
|  |  |  |

**Table of Contents**

[Game Overview 3](#_TOC_250023)

[Game Summary 3](#_TOC_250022)

[Platform 3](#_TOC_250021)

[Development Overview 3](#_TOC_250020)

[Development Team 3](#_TOC_250019)

[Development Environment 3](#_TOC_250018)

[Development Hardware 3](#_TOC_250017)

[Development Software 3](#_TOC_250016)

[External Code 3](#_TOC_250015)

[Game Mechanics 4](#_TOC_250014)

[Main Technical Requirements 4](#_TOC_250013)

[Architecture 4](#_TOC_250012)

[Game Flow 4](#_TOC_250011)

[Graphics 4](#_TOC_250010)

[Audio 4](#_TOC_250009)

[Artificial Intelligence 4](#_TOC_250008)

[Networking (If Applicable) 5](#_TOC_250007)

[Physics (If Applicable) 5](#_TOC_250006)

[Game Objects and Logic 5](#_TOC_250005)

[Data Management and Flow 5](#_TOC_250004)

[User Interface 5](#_TOC_250003)

[Game Shell 5](#_TOC_250002)

[Play Screen 6](#_TOC_250001)

[Technical Risk 6](#_TOC_250000)

# Game Overview

This section provides an overview of the game, suitable to someone new to the project.

## Game Summary

Project BlightBrew is an experience focused on the magical experience of becoming a master alchemist while repairing and running a potion shop tucked away into a magical region of unknown wonder. The player must solve puzzles, explore, craft potions and build their reputation to become a successful store owner and craftsman, all while combating a parasitic blight that threatens the region.

## Platform

PC

# Development Overview

This section discusses key aspects of the development of the game, as opposed to the game itself.

## Development Team

* Carson – Artist
* David – Artist | Design Contact
* Isaac – Artist
* Jiaming – Artist | Art Contact
* Perrin – Programmer | Programming Contact
* Ken – Artist | Comms Contact
* Xander – Programmer
* Xavier – Artist

## Development Environment

This describes the environment used to develop the game. In theory, someone could set up an environment like what is described here and everything needed to work on the game.

### Development Hardware

Development level computers running Windows 10/11

### Development Software

* Assembla – Source Control & Agile Development / Scrum
* 3DSMax 2023 – 3D Modelling
* MudBox – Modelling
* ZBrush - Modelling
* Visual Studio 2022 – Programming
* Unreal Engine 5.0 - Game Engine

### External Code

Unreal Engine 5.0 (Source + API)

A discussion of the content pipelined

used to bring art and sound assets into the game would be useful as well.

# Game Mechanics

This is the main part of the document. This goes through the architecture and implementation notes on the main subsystems and control elements of the game.

## Main Technical Requirements

This provides a description of the key technical requirements imposed by the game’s design. This will include critical functionality as well as constraints on this functionality (like performance, and so on). Some of these issues can be discussed in the other subsections under Game Mechanics.

## Architecture

Graphical user interface

Description automatically generated

The Game’s Architecture follows the above model. The Player is the central system in which all of the modules interact with each other through.

* Player system - the system is our core system and is what the player interacts with to use all our other systems.
* Locomotion system - an advanced movement system we’re using to give us more control over the players movement. This gives us variables we can control, such as stamina, and additional functionality, such as climbing and vaulting.
* Potion system – a system meant to be easy to use for designers. The system allows for the designers as much freedom as possible, while giving the coders a minimal amount of updating to do. This is the main system the player will use to complete puzzles and generate income.
* Inventory system – a system meant to be easy to use for designers. The system allows for the designers as much freedom as possible, while giving the coders a minimal amount of updating to do. The system works as a container to hold items that the player collects. This includes the players backpack, as well as storage containers around the world.
* Resource system – a system meant to be the first of the steps for the player to generate income. This system gives the player ways to harvest materials and shows off behavioural logic for when the system is used.
* Crafting System – the system in between the Resource system, and the Shop/Potion systems, in terms of actual gameplay. This systems job is to refine the resources the player gathers into other items or potions for use in one of the other systems.
* Shop system - a system to generate income, and allow for the player to improve. The system interfaces with AI, the player, and Storage objects, to give the player a way of selling items to NPC’s. This can have numerous effects on the AI and is the primary source of income, a necessary resource for improving the players arsenal, and serves to break the monotony of just grinding for resources.
* AI System - The system dealing with the NPC’s in the world, including fauna. The system interfaces with the shop system, allowing for NPC’s to be customers, as well as gives the AI their logic for movement, interaction, and anything else.
* Economy System – A system that goes unseen by the player, changes the value of items based on selling history, value of the local economy, and a degree of randomness. This should make the world feel a little bit more alive, and give the player a reason to continually adjust prices based on customer responses.
* User Interface System – A system used to show, hide, and allow the player to control the Interface. The interface controls almost every system in some way, but mainly through the player.

## Game Flow

*This section provides a discussion of how control flows through the game. This includes a*

*description of the main control loop of the game and how it interacts with the various parts of the game, as well as the various different states that the game can be in (title screen, menus, playing, paused, and so on, perhaps with sub-states). Again, diagrams should be provided.*

**Main Control Loop:**

* Player can Move around, and Vault/Climb the environment.
* Player can Sneak, Jump and Sprint
* Player can Pickup: Potions, Ingredients, Items
* Player can Throw: Potions
* Player can Consume: Potions, Ingredients
* Player can navigate QuickSlots to choose potions to consume/throw
* Player can open backpack to access inventory & equip items to QuickSlots

**Game States:**

* Main Menu
* Paused
* Playing

## Graphics

*This section discusses the 3D/2D graphics elements of the game. Focus on the core aspects as opposed to interface elements, as these will be discussed later. (This would include rendering, sprite management, and so on.) If there are any constraints on the type of content that can be used here, it should be discussed.*

The game is in 3D and is rendered in 3D. All game objects have 3D meshes. All UI is 2D, and either rendered in screen space or world space.

## Audio

*This section discusses the sound elements of the game, including music playback, sound effect playback, and so on. If there are any constraints on the type of content that can be used here, it should be discussed.*

None yet.

## Artificial Intelligence

*This section discusses the control of all non-player active entities in the game. Identify how this control is done (scripting, state machines, rules, and so on) and provide details for each entity.*

None yet.

## Networking (If Applicable)

*If there are online features to the game, they should be discussed here. This includes a description of protocols used, and other technical details.*

N/A

## Physics (If Applicable)

*If physical simulation is used, for collision detection, object interactions, and so on, a discussion of what is done and how is provided here.*

Physics are used for:

* Throwing Potions
* Locomotion (Walking/Sprinting, Jumping, Swimming, Climbing, Vaulting)
* Collision Detection (for potions)
* Raycasts for interactions (player interacting with game objects)
* Raycasts for focus (player looking at game objects)

## Game Objects and Logic

*This is a discussion of all gameplay objects and logic used to support the game, as opposed to the various core subsystems discussed above. (One way of thinking about it is that these are all the game specific elements to your game … the other aspects described in the subsections above could theoretically be provided by an engine, and might be, depending on your game.)*

*This should include subsections for each main game object or set of objects.*

Player:

* Can Locomote around the Map (Walk/Sprint, Jump, Crouch, Swim, Climb, Vault)
* Can Pickup Ingredients
* Can Pickup Potions
* Can Throw/Consume Potions
* Can Consume Ingredients
* Can Interact with InteractableActors
* Can Focus on InteractableActors

**Locomotive Game Objects:**

* Any Actor with a Static Mesh
* Has been tagged either “Vaultable or Climbable”

**Ingredients:**

* InteractableActor
* OnInteract -> Add to Instigator’s Inventory
* OnFocus -> Display info to Instigator (such as Name & Value)

**Potions:**

* Throwable and/or Consumable

## Data Management and Flow

*Game data will need to be loaded and saved at various points in time. This section should describe the procedures used to do loading, saving, and any processing of this data (such as compression/decompression, encryption/decryption, parsing, and so on), as well as the various different types of files used to contain this information.*

Not Implemented Yet.

# User Interface

This section describes how information is presented to the player, and how input is received in return. This includes display layout, feedback, controls, and so on.

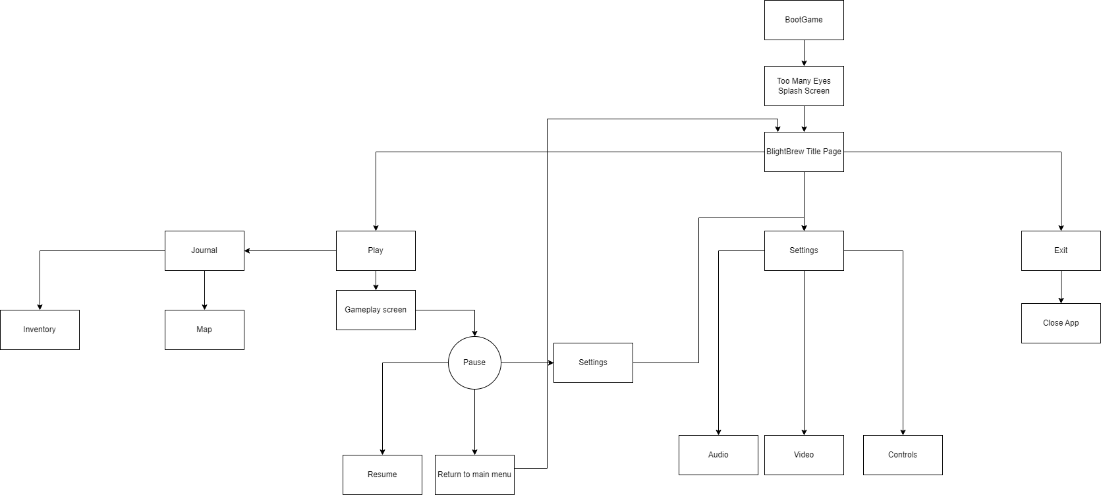
## Game Shell

*This discusses all screens presented to the player outside of game play. If there will be sound,*

*music, or tactile feedback in any of these screens, it should be discussed. Controls should also be discussed to describe how screens are changed or how interface elements (buttons, etc.) are manipulated by the player.*

*The game will likely have menus given to the player to control the flow of out-of-game activities. This includes setting game options, controlling game flow (starting, stopping, pausing, and so on), and a variety of other tasks. The various menus must be identified, as well as the flow between them, and circumstances under which they appear and disappear.*

*Additionally, other screens may be presented to the player. This includes splash screens with logos and/or legal notices, credit screens, and so on. These should also be discussed.*



Game Shell / Screen Flow:

* BootGame – The initial screen for the Game
  + Transitions to SplashScreen
* SplashScreen – The Splash Screen for when the game is loading. This screen will display the company Logo & Name, Algonquin College Logo & Name, Unreal Engine 5 Logo & Name
  + Transitions to TitlePage
* TitlePage – The First Input enabled UI screen the player will see. This will display the project name, logo, and navigation options:
  + Play – Input button
    - Play Transitions to GamePlayScreen & enables input for PauseScreen and Journal
  + Settings – Input Button
    - Settings Transitions to SettingsScreen
  + Exit – Input button
    - Exit exits the game / close app
* Settings – The screen where a player can use various inputs & widgets to modify game settings. There are 3 sub screens in settings
  + Audio – Input button
    - Toggles the Audio SubScreen
  + Video – Input button
    - Toggles the Video SubScreen
  + Controls – Input button
    - Toggles the Controls SubScreen
* Play – Shows the PlayScreen (HUD), and has input enabled to open the Journal and PauseScreen
  + Transitions to PauseScreen
  + Transitions to Journal
* PauseScreen – Pauses the game & input, displays three buttons for input:
  + Resume – Input button
    - Resumes play
  + Settings – Input button
    - Transitions to SettingsScreen
  + Quit to Main Menu – Input button
    - Transitions to TitlePage
* Journal – Displays a screen on top of the Play Screen, that looks like a book. This screen contains TWO (subject to change) submenus:
  + Map – Displays the Map of the current area
  + Inventory – Displays the Player’s current inventory & QuickSlots. Allows the player to move items around, and move items between inventory & QuickSlots.

Controls:

* J – Toggle Journal
* M – Toggle Map (SubMenu of Journal)
* B – Toggle Inventory (SubMenu of Journal)

## Play Screen

The Hud:

* Time Widget – Shows time of day, portion of day (morning/afternoon/etc), and a progress bar showing how far the day has progressed so far.
* QuickSlots – Shows the currently equipped potions, the quantities, and a highlight around the QuickSlot that is currently selected.

Controls:

None. The HUD will show during the Playing Game State

# Technical Risk

**Failing to create a satisfying bridge between exploration and store management**

* Build base experience (explore > find resources > find recipe > craft > sell)

**Shortage of programmers**

* Use abundance of artists to take on most of the workload
* Few programmers will build tools for artists
* Artists will use tools to take on most “in-engine” implementation of mechanics
* Scalability
* Designers and artists will use world, level and puzzle design to create variety based around this simple, scalable foundation
* Allows us to manage the scope of the project as necessary