GUI Programming  
Potion Shopkeeper

short line

Zora Jane Kerr  
17th September, 2023

Contents

[Document purpose 2](#_Toc145869514)

[Game requirements 3](#_Toc145869515)

[Device specifications 3](#_Toc145869516)

[Desktop 3](#_Toc145869517)

[Mobile 3](#_Toc145869518)

[Browser 3](#_Toc145869519)

[Controller inputs 4](#_Toc145869520)

[Software assumptions 4](#_Toc145869521)

[Features 4](#_Toc145869522)

[1. The game world 4](#_Toc145869523)

[2. Game objects 4](#_Toc145869524)

[2.1 Player 4](#_Toc145869525)

[2.2 Laboratory 4](#_Toc145869526)

[2.3 Villager 4](#_Toc145869527)

[2.4 Items 5](#_Toc145869528)

[3. User interfaces 5](#_Toc145869529)

[3.1 Player 5](#_Toc145869530)

[3.2 Laboratory 5](#_Toc145869531)

[3.3 Villager: 6](#_Toc145869532)

[3.4 Main menu: 6](#_Toc145869533)

[4. Advancing in-game time 7](#_Toc145869534)

[4.1 ‘End Day’ function 7](#_Toc145869535)

[UI Wireframe Mockups 8](#_Toc145869536)

[Scene 1: Game world UI (partially implemented) 8](#_Toc145869537)

[Scene 2: Picking up Ingredients (not implemented) 8](#_Toc145869538)

[Scene 3: Using the Laboratory UI (partially implemented) 9](#_Toc145869539)

[Scene 4: Using the Villager UI (not implemented) 10](#_Toc145869540)

[Scene 5: Main menu UI (partially implemented) 11](#_Toc145869541)

[Scene 6: End Day UI (not yet implemented) 12](#_Toc145869542)

[Testing report 13](#_Toc145869543)

[Feedback from reviewer 13](#_Toc145869544)

[Test table 13](#_Toc145869545)

# Document purpose

This is the GUI design document for the game *Potion Shopkeeper*. The player is the potion maker of a small village and the game takes place in their house and garden. The player gathers ingredients from their garden, combines those ingredients in their laboratory to make potions, and gives those potions to villagers who present with varying problems. The player must experiment with different ingredients to make different potions which address the needs of the villagers.

Potion Shopkeeper is a relaxed, 1-player, 3D side-scrolling point & click game. The game is a simplified version of *Potion Craft: Alchemist Simulator*.

Potion Shopkeeper will be developed in Unity (2021.3.13f1) with Visual Studio 2022 with release builds for 3 platforms:

* standalone PC application (Windows 10).
* web browser (Chrome).
* mobile phone (Android App Bundle).

This document records the GUI design for:

* The gameplay loop.
* The Main and Settings menus.

**NOTE: This document includes material intended to support Cross-Platform Development. The only GUI implementations intended to be demonstrated in-engine for the purposes of *User Interface Assessment Task 3 – Design and Implement a Graphical User Interface* are:**

* **Menu minimise / maximise on button click**
* **Menu slider bar setup**
* **Menu boolean toggle**
* **Inventory drag & drop functionality**

# Game requirements

## Device specifications

The game is anticipated to require the lowest levels of computational demand, and so the minimum device specifications are identical to the [minimum specifications required to run Unity 2021.13](https://docs.unity3d.com/2021.3/Documentation/Manual/system-requirements.html#mobile).

### Desktop

|  |  |
| --- | --- |
| Operating system | Windows 7 (SP1+), Windows 10 and Windows 11 |
| Version | x86, x64 architecture with SSE2 instruction set support. |
| CPU | ARMv7 with Neon Support (32-bit) or ARM64 |
| Graphics API | DX10, DX11, DX12 capable. |
| Additional requirements | Hardware vendor officially supported drivers. |

### Mobile

|  |  |
| --- | --- |
| Operating system | Android |
| Version | 5.1 (API 22)+ |
| CPU | ARMv7 with Neon Support (32-bit) or ARM64 |
| Graphics API | OpenGL ES 2.0+, 3.0+, Vulkan |
| Additional requirements | - 1GB+ RAM  - Supported hardware devices must meet or exceed Google’s Android Compatibility Definition (Version 9.0) limited to the following Device Types:  1. Handheld (Section 2.2)  2. Television (Section 2.3)  3. Tablets (Section 2.6)  - Hardware must be running Android OS natively. With the exception of Android for Chrome OS, Android within a container or emulator isn’t supported. |

### Browser

|  |  |
| --- | --- |
| Operating system | Windows, macOS, and Linux |
| Version | Workstation and laptop form factors. |
| CPU | Versions of Chrome, Firefox, Safari or Edge (Chromium-based) that are:  - WebGL 1.0 or 2.0 capable  - HTML 5 standards compliant  - 64-bit  - WebAssembly capable  Note: WebGL 1.0 support is deprecated. |

## Controller inputs

PC and browser release builds will be configured for mouse and keyboard inputs as well as touchscreen, and mobile phone will only be configured for touchscreen.

## Software assumptions

Potion Shopkeeper has no software dependencies other than those presumed to be running on the operating system the user plays the game on. There are no interfaces between the game and other applications, and only the web browser publication requires an internet connection.

# Features

There are 4 key features to the game which require UI elements or visual effects, as follows:

1. The game world
2. Game objects
3. User interfaces
4. Advancing in-game time

## The game world

The game world is a simplified 3D model of a house and a garden, navigated on a 2D plane by player inputs (left, right, up, down). The scene itself cannot be interacted with and is a background. A perspective camera is locked to the player, and UI elements are screen-space overlays.

## Game objects

### Player

* + - The player’s transform in the 3D game world must visually update in response to user inputs (keyboard, mouse, touchscreen).

### Laboratory

* + - The laboratory’s transform cannot move in the game world but the object can be interacted with through mouse clicks or touchscreen presses.

### Villager

* + - The villager’s transform cannot move in the game world but can be interacted with through mouse clicks or touchscreen presses. Villager visibility will be conditional on whether the front door is open or closed.

### Items

* + - Items have a transform in the game world which must visually update in response to user inputs (drag/drop, double-click).
      * **Ingredients**: Ingredient objects must be able to be dragged and dropped between:
        + the game world and the player’s inventory; and
        + the player’s inventory and the laboratory.

Ingredients will be visually highlighted in the game world in response to mouseover and will make a sound in response to dragging and dropping, such as a rustling grass sound.

* + - * **Potions**: Potion objects must be able to be dragged and dropped between:
        + the laboratory and the player’s inventory; and
        + the player’s inventory and the inventory of a villager.

Potions will be visually highlighted in the game world in response to mouseover and will make a sound in response to dragging and dropping, such as a liquid sloshing sound.

## User interfaces

### Player

* + UI 1: Inventory button

A button in the top-right corner of the screen overlay which minimises / maximises the Player inventory UI.

* + UI 2: Inventory

A UI representing a 9-slot container which may hold both Ingredient and Potion type objects, with an explanatory title bar and a button to minimise/maximise its visibility on the screen.

Player inventory defaults to being on the right-hand-side of the screen.

### Laboratory

Laboratory UIs will default to being on the left-hand side of the screen.

* + UI 1: Inventory (the ‘Cauldron’)

A UI representing a 3-slot container which may only hold Ingredient type objects, it has an explanatory title bar and a “Brew” button.

This UI does not have minimise/maximise buttons, it is accessed by clicking its associated game object in the world, and closed by clicking anywhere else in the game world while the UI is open.

* + UI 2: Completion screen

A UI representing a 1-slot container which may only hold Potion type objects and appears in response to the Cauldron being used successfully.

This UI does not have minimise/maximise buttons, it appears in response to is accessed by clicking its associated game object in the world, and closed by clicking anywhere else in the game world while the UI is open.

### Villager:

* + UI 1: Inventory

A UI representing a 1-slot inventory which may only hold Potion type objects. This UI does not have minimise/maximise buttons, it is accessed by clicking its associated game object in the world, and closed by clicking anywhere else in the game world while the UI is open.

* + UI 2: Completion screen

A UI representing a menu with a success message and a button which closes the menu.

This UI does not have minimise / maximise buttons, it appears in response to a correct Potion being given to a Villager. It can be closed by clicking its button or anywhere else in the game world while the UI is open.

### Main menu:

* + UI 1: Options button

A button in the top-left corner of the screen overlay which minimises / maximises the Mani menu UI.

* + UI 2: Main menu UI

A UI representing a menu with x buttons:

1. Resume
2. Settings
3. Quit game

## Advancing in-game time

### 4.1 ‘End Day’ function

* + UI 1: End Day button

A button with text which may finalise the current ‘day’. This refreshes the Ingredient items in the garden and refreshes the Villager who needs a Potion.

* + UI 2: End Day confirmation UI

Clicking the “End day” button will make visible a UI prompt for the user to confirm whether they really want to end the current day. If they confirm this by clicking ‘Yes’, the screen will fade to black, play a different musical score, and the Player’s transform will be reset to their start position.

# UI Wireframe Mockups

## Scene 1: Game world UI (partially implemented)

|  |  |
| --- | --- |
| A screenshot of a computer screen  Description automatically generated | This is a miniature abstraction of the 2D side-scrolling view of the 3D game world environment.  Red: Villager object.  Green: Laboratory object.  Blue: Player object.  Orange: Ingredient objects (pickups).  The wireframe structure represents the player’s house, with a demarcation between the house and the garden (right-hand side of the screen). |

## Scene 2: Picking up Ingredients (not implemented)

|  |  |
| --- | --- |
| A screenshot of a computer  Description automatically generated | Ingredients can be moved from the game world to the player’s inventory by double-clicking (whether inventory is open or not) or by dragging and dropping (inventory must be ‘open’). The Player object will navigate near to the Ingredient before the ‘pick-up’ occurs.  This screen shows an Ingredient being moved from the game world to the inventory by dragging and dropping while the inventory is open. |

## Scene 3: Using the Laboratory UI (partially implemented)

|  |  |
| --- | --- |
| A screenshot of a computer  Description automatically generated  **A screenshot of a computer  Description automatically generated**  A screenshot of a computer  Description automatically generated  A screenshot of a computer  Description automatically generated | **Enter**:  When the player clicks on the Laboratory object in the game world, the Player object will navigate near to it and 2 UIs will open: the Laboratory UI (LEFT) and the Player Inventory UI (RIGHT). Both UIs will be overlaid atop the 3D game world which will be visible behind them.  **Exit**: The Laboratory UI can be exited at any time by clicking anywhere other than one of the UIs. All UIs will close.  **Interactivity**:  The player can drag & drop / double click Ingredient type items in their own inventory to move them to the Cauldron.  **Functionality**:  Once 3 Ingredient-type objects have been inserted into the Cauldron, the Brew button of the Cauldron UI will become illuminated and may be clicked.  Different 3-Ingredient combinations are pre-defined in the program (‘recipes’) and each creates a different Potion item.  If the Brew button is clicked, and the 3 combined Ingredients represent a Potion recipe, the Ingredients are destroyed and the Laboratory Completion Screen UI opens, showing which Potion has been created. The player may either double click the Potion from the Completion UI to add it to their inventory, drag and drop it into the inventory, or exit the Laboratory to automatically move it into their inventory. |

## Scene 4: Using the Villager UI (not implemented)

|  |  |
| --- | --- |
| A screenshot of a computer  Description automatically generated  A screenshot of a computer  Description automatically generated  *Success state:*  A screenshot of a computer  Description automatically generated  *Fail state:*  A screenshot of a computer  Description automatically generated | **Enter**:  When the player clicks on the Villager object in the game world, the Player object will navigate near to it and 2 UIs will open: the Villager UI (LEFT) and the Player Inventory UI (RIGHT). Both UIs will be overlaid atop the 3D game world which will be visible behind them.  **Exit**: The Villager UI can be exited at any time by clicking anywhere other than one of the UIs. All UIs will close.  **Interactivity**:  The player can drag & drop / double click a Potion type item in their own inventory to move it to the Villager.  **Functionality**:  Once 1 Potion-type object has been placed in the Villager inventory UI, the Sell button of the Villager UI will become illuminated and may be clicked.  ***Success state:***  If the Sell button is clicked and the Potion is the correct Potion required by the Villager, the Potion is destroyed and the Villager Completion Screen UI opens with a grateful message.  ***Fail state:***  If the Sell button is clicked and the Potion is not the correct Potion required by the Villager, a shake effect will trigger on the UI, the Villager Completion Screen UI opens with a message, and the Potion will be moved back to the Player’s Inventory. |

## Scene 5: Main menu UI (partially implemented)

|  |  |
| --- | --- |
| A screenshot of a computer  Description automatically generated  A screenshot of a computer  Description automatically generated | **Enter:**  When the player clicks on the Options button in the top-left corner of the screen, the Main menu UI will appear as a screen overlay and the game world opacity behind it will rise to obscure the scene.  **Exit**: The Main menu UI can be exited at any time by clicking the Resume button or anywhere other than the UI.  **Interactivity**:  The player can click one of the 3 Main menu UI buttons: Resume, Settings, Quit game, the screen overlay Options button, or anywhere other than the UI.  **Functionality (Main menu UI)**:   1. Resume button: Return to the scene. 2. Settings button: Advance to the Settings UI. 3. Quit game button: Quit the game. 4. Options button: Return to the scene. 5. Click anywhere else: Return to the scene.   **Functionality (Settings UI)**:   1. Music slider: Control master music volume. 2. FX slider: Control effects volume. 3. Key bindings button: Change input bindings. 4. Resolution options form button: Change resolution. 5. Language options form button: Change language. 6. Back button: Go to the Main Menu UI. |

## Scene 6: End Day UI (not yet implemented)

|  |  |
| --- | --- |
|  | **Enter:**  When the player clicks on the End Day button in the top-middle of the screen, the End Day Confirmation UI will appear as a screen overlay and the game world opacity behind it will rise to obscure the scene.  **Exit**: The End Day Confirmation UI can be exited at any time by clicking the No button or anywhere other than the UI.  **Interactivity**:  The player can click one of the 2 End Day UI buttons or click anywhere other than the UI.  **Functionality (Main menu UI)**:   1. Yes button: Advance the Day. 2. No button: Return to the scene. 3. Click anywhere else: Return to the scene. |

# 

# Testing report

## Feedback from reviewer

From Michael Burford:



## Test table

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected result | Outcome | Comments |
| Main menu opens and closes | Main menu UI minimises / maximises on mouse click of the Options button. | Works correctly. | Nil. |
| Settings menu opens and closes | Settings menu UI minimises / maximises on mouse click of the Settings button within the Main Menu UI. | **First pass**: Partially successful; Settings menu UI opened but Main menu UI also remained open behind it, and Settings remains open when clicking the Options button again.  **Second pass**: Works correctly. | Needed to add an OnClick event to the Settings button in the Main menu UI which toggles the Main menu, and an OnClick event to the Options button to turn off the Settings UI. |
| Settings > Resume closes the UIs | Close the Main menu UI. | Works correctly. | Nil. |
| Settings > Quit game exits | Close the application. | Works correctly. | Add a confirmation screen before exiting for Cross-Platform prototype |
| Main menu children update properly | 1: Slider Input Fields are updated in real time when the connected slider bar is moved.  2: Toggle box updates connected boolean in SettingsPalette.  3: Main menu UI elements update with back-end changes (SettingsPalette) when maximising. | 1: Works correctly.  2: Works correctly.  3: Works correctly. | Nil. |
| Player inventory opens and closes | Player inventory opens and closes in response to mouse click on Inventory button | Works correctly | Not yet implemented as per the design doc (a change for the prototype – works in principle). |
| Player and Laboratory 3D objects have a mouseover glow | Player and Laboratory greybox objects have a glowing edge when moused over and clicked | Works correctly. | I downloaded and used the Unity Store asset [Quick Outline](https://assetstore.unity.com/packages/tools/particles-effects/quick-outline-115488) by Chris Nolet, with accompanying code by [DA LAB](https://www.youtube.com/watch?v=qYnAkMGbgwo). |