GUI Programming  
Potion Shopkeeper

short line

Zora Jane Kerr  
14th September, 2023

# Table of Content

[Table of Content](#_Toc144995399)

[Purpose](#_Toc144995400)

[Requirements](#_Toc144995401)

[Features](#_Toc144995402)

[1. Game world](#_Toc144995403)

[2. Manipulate game objects](#_Toc144995404)

[2.1 Player](#_Toc144995405)

[2.2 Laboratory](#_Toc144995406)

[2.3 Villager](#_Toc144995407)

[2.4 Items](#_Toc144995408)

[3. Open / close user interfaces](#_Toc144995409)

[3.1 Player](#_Toc144995410)

[3.2 Laboratory](#_Toc144995411)

[3.3 Villager:](#_Toc144995412)

[4. Advance in-game time](#_Toc144995413)

[4.1 ‘End day’ function](#_Toc144995414)

[UI Wireframe Mockups](#_Toc144995415)

# 

# Purpose

This is the GUI design document for the game *Potion Shopkeeper*. The game takes place in the house and garden of the player, a potion maker of a small village, who 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 cosy, 1-player, non-competitive 2D sidescroller in a 3D game world. Similar games include *Potion Craft: Alchemist Simulator* and *Spellcaster University*.

This document records the GUI design for only the main gameplay loop, and not all possible scenes in the game.

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).

# Requirements

## Specifications

Minimum system specs: Must run on a PC costing less than AUD$750 as at 31 December 2023.

[Insert RAM, CPU, PSU, GPU minimum spec when known]

## 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

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:

## 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

### 2.1 Player

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

### 2.2 Laboratory

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

### 2.3 Villager

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

### 2.4 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.

* + - * **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.

## Open / close user interfaces

### 3.1 Player

* + 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 is generally on the right-hand-side of the screen.

### 3.2 Laboratory

Laboratory UIs will generally be 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.

### 3.3 Villager:

* + 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.

## Advance in-game time

### 4.1 ‘End day’ function

* + A button with text which finalises the current ‘day’. This refreshes the Ingredient items in the garden and refreshes the Villager who needs a Potion. Clicking the button will prompt the user to confirm whether they do 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

|  |  |
| --- | --- |
| A diagram of different colored cubes  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). |

On mouse-over

On double-click

On drag / drop

Scene 2: Picking up Ingredients

|  |  |
| --- | --- |
| A diagram of a graph  Description automatically generated with medium confidence | 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

|  |  |
| --- | --- |
| A diagram of a diagram  Description automatically generated  **A diagram of a diagram  Description automatically generated with medium confidence**  A screenshot of a computer screen  Description automatically generated  A screenshot of a computer screen  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. |

# 

Screen 4: Talking to a Villager

|  |  |
| --- | --- |
| A screenshot of a computer  Description automatically generated  A screenshot of a computer screen  Description automatically generated  A screenshot of a computer screen  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.  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 and the Potion will be moved back to the Player’s Inventory.  If the Potion was correct, the Potion is destroyed and the Villager Completion Screen UI opens with a grateful message. |

|  |
| --- |
| Code Snippet:  function update()  {  if (isKeyDown(LEFT\_KEY) || isControllerButtonPressed(0, 6))  {  marioXPos -= 1;  }  if (isKeyDown(RIGHT\_KEY) || isControllerButtonPressed(0, 7))  {  marioXPos += 1;  }  } |