Technical University of Denmark

02823

Introduction to Computer Game Prototyping

Mouse Potion Game Prototype

Authors:	*Student Nr.:*
Catrin Hathaway Murphy	s191502
Louise van der Peet	s191843
Monika Frolcova	s182194
Riley Goodling	s191362

December 16, 2019

Contents

1 Introduction

Our concept for this game started with a simple idea: You are a mouse that has been cursed by a witch, and through exploring the witch's magic garden the mouse is able to regain its memories and eventually turn itself human again. From this concept, we expanded the game into it's current version. In the game, you explore a garden whose main layout remains the same throughout each level, while you craft potions at the end of each level to either further the story or to unlock access to further parts of the garden. The main goal of each level is to find a specific list of ingredients for each potion, and to do this the player must solve puzzles and find hidden items throughout the level. Currently, the game contains one level with three ingredients to collect, as well as most of the infrastructure to support easily adding additional levels.

When making the game, we were inspired partly by the metroidvania genre in terms of creating a map that remains similar but evolves over time, and creating a world that slowly opens up for further exploration. In its current state, the game also shares a lot of similarity with point-and-click adventure games in terms of the style of many of its current puzzles, which often involve using items in the correct locations to find ingredients or unlock new items.

2 Game Description

2.1 Rules

The rules follow those of a simple adventure game. The player controls the mouse using WASD and picks up items using F. There is no jump, as this is a top down 2D game. In order to finish a level, the player needs to collect all the required ingredients for the specified potion. There is no dying or health in this game, only the need to complete the level. The required ingredients will be hidden around the map, and most can only be accessed by solving some sort of puzzle. The mouse has an inventory of items in its backpack, which the player can see by pressing the I key and rotate between inventory items using Z. By pressing M, the player can see the map in order to have a better idea of objectives and puzzle elements. There are bonus items around the map as well, but access to those is dependent on the level and cannot be reached in Level 1.

2.2 Desired features

At the start of the project we wrote a list of all the features the game should have. The list looked somewhat like this.

- Core game-play: the character should try to collect a different list of items (or ingredients) in each level. There will be a top-down puzzle exploration to

find the items. There will be a similar garden layout throughout the levels, but differences in the environment as time passes. As skills are added to the character, other parts of the garden will open up.

- Potions can be used to continue the plot, gain skills and possibly for other uses.
- More hidden items can be found, which possibly allow for small bonuses or maybe even to craft extra potions.
- Bonus potions you create can later give you additional skills or ability to access bonus areas.
- The game could possibly contain some platforming. For instance, when you go inside to make the potions, to be able to create the potion you might need to do a platform level.
- We could add a time element to solve the puzzles in a certain amount of time.

2.3 Technical challenges

As for technical challenges, we did not encounter many. The prototype we made was in our scope and all problems encountered could be easily solved by googling since the level we made is quite basic. If the game was to be developed further we might run into more technical challenges, due to there having to be more complicated mechanics than in a prototype level.

3 Implementation

At first we will describe the overall structure of the game and in following sections we will delve deeper into the design of the puzzles, scene and the implementation of game mechanics.

3.1 Level design

When the game starts a player will see the initial level selection scene (Fig 3.1). In current prototype only Level 1 has been implemented so far, however in future the player will be able to unlock further levels and resume the game at a selected level. In Unity, we have designed the game to have one scene for each level, as well as additional scenes for crafting and cutscenes.

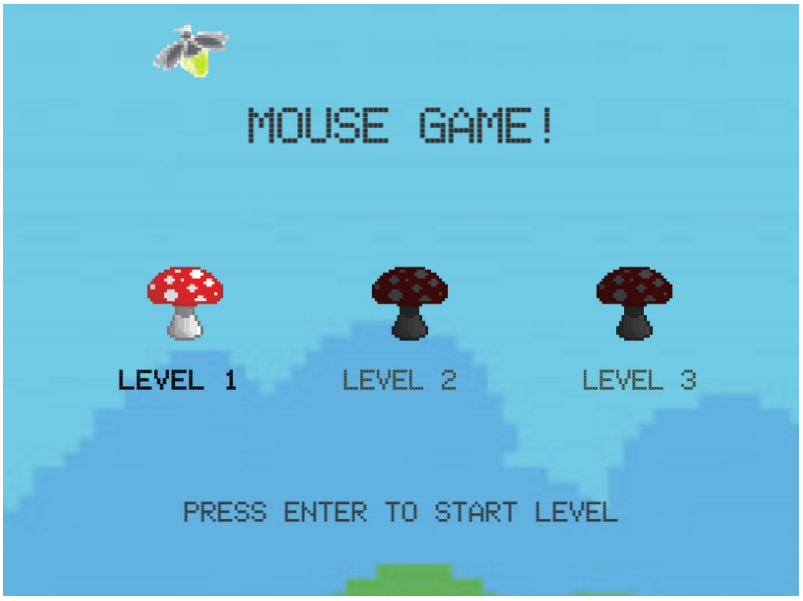


Figure 3.1: Level selection Scene

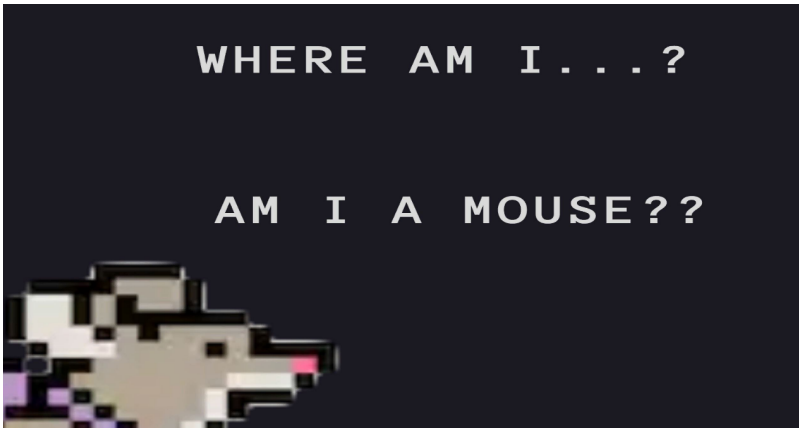


Figure 3.2: Level 1 Cut Scene

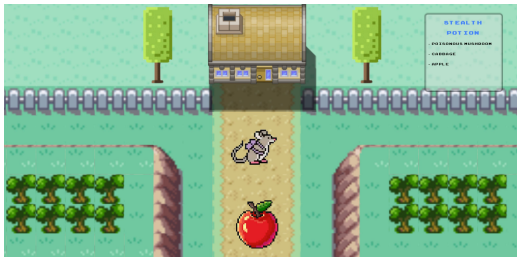


Figure 3.3: The first level

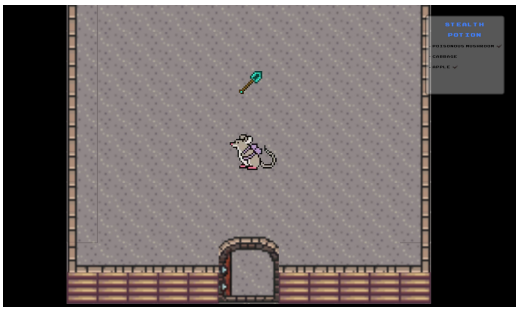


Figure 3.4: The first level: hidden shed area

Figure 3.5: Potion Creation Cut Scene

Figure 3.6: Potion Creation Cut Scene

3.2 Puzzle design

We wanted our game to be based around environmental puzzles. When coming up with the puzzles, we had many ideas to hide things around the garden and in surrounding areas. Due to the time constraint, we were only able to include one level, with some of the ideas that we had for puzzles.

Our general philosophy in adding the puzzles was to create variety of puzzles that either involved observing key areas in the environment or finding interesting ways to interact with the environment. We wanted the puzzles containing required items not to be impossibly difficult, and to contain hints to guide the player towards the solution. Then, the possibility of adding bonus items can be used to add more difficult puzzles or even just placed in an area to signal that for now the player is done with that area.

3.2.1 The Apple

The apple, while technically a required ingredient, exists mainly as a sort of tutorial to show the player how they will find items and ingredients, and how to pick them up. While we were undecided on if we wanted a tutorial level, we decided for the moment to use the in-game UI system in order to implement some simple tutorial pointers such as this.

3.2.2 The Mushroom

Some of the puzzles were not meant to be extremely complex, like finding the mushroom. We thought the game would be best if the puzzles were not all extremely

complicated, but more of a unique puzzle that requires observation and trial-and-error skills. The mushroom is hidden behind an opaque bush, which becomes visible when the mouse walks into it. In making this puzzle, we hoped that the fact that the path continues to lead into the bush would hint well enough to players that there was something there. In play tests this has mostly been true, but it is difficult to know if the difficulty is in the right place without a larger sample. This puzzle was initially conceived as one with a bonus item, but to prevent the map from being too large in this initial level we chose to instead make it an ingredient puzzle.

3.2.3 The Firefly/Cabbage Puzzle

For the firefly puzzle, we wanted to add a multi-step puzzle that would take some thinking and observation from the player. The player first has to collect a stick and some yarn from the map in order to combine them to make a net. Then, the player must catch the three fireflies which have been flying around the garden and place them back into a jar. To help signal that this is the goal, when the player first sees the jar, one of the fireflies flies out of the jar. While this isn't always noticeable it was hoped that this would help to offset the fact that the jar is not super clearly a jar. Once the fireflies are in the jar, the door to the shed opens and the player can grab a shovel which they use to pick up the cabbage.

3.2.4 The Cat

While the cat isn't a puzzle element that needs to be solved in this level, it was an element commonly mistaken as part of the puzzle, so we will discuss it here. The cat was placed in the level as one of the points of future exploration, so that in the future the player can find some way past the cat. However, many players instead thought that the stick could be used to kill the cat, or something similar. While to some degree this is fine - when it doesn't work the player can move on and work on different puzzles, many players fixated on this cat as the solution. As such, this element may need some work to better indicate to players that it isn't the correct path, possibly by placing a bonus star within the cat area which must be reached by luring the cat far enough to run to the star. This could then indicate to the player that the cat puzzle is solved for the time being.

3.2.5 Future Puzzles

For future levels, we have many ideas for puzzles that have similar skills needed to solve them. We tried to use aspects of a garden (like bugs and a garden shed) to fold into our puzzles. A puzzle that relates to outer space or pirates would not be a great fit for our game. We also tried to make players see things around the map that they could believe is important to a puzzle. We have already added some of the foundations for future puzzles in the map. We added the cat chasing the player as a way to make the player believe that something important is in the cat's area. We also added some stars around the map that could come into play in future levels, but have no purpose in the current level 1. In addition, many of the scenic elements (such as the water blocking one border, and the fence itself) can be changed to provide access to further areas of the map.

3.3 Scene design

The tilemap was made from tiles a Pokemon tile set found on DeviantArt. (1). The level is bound by fences, water and the cat, so the player cannot possibly walk out of the level. We used trees, bushes, flowers and cat items to decorate the scene. The shed is meant for further levels, as well as the visible bonus stars. The apple, cabbage patch, mushroom, stick, fireflies, jar and yarn are usable items meant to progress in the level.
* tilemaps * assets
The cat guards a section of the map the player is not supposed to go to yet. It moves towards the player slowly when the player reaches a certain point ($y < -12$). When the player goes too far ($y < -20$), the cat will run at it. When the cat reaches the player, it dies and the player is put back at the starting position. All of this is written in the CatMove script. It works by binding a wayPoint game object to the player. This keeps track of the current coordinates of the player so the cat script can access it. In future levels, the player might be able to distract the cat or make

a sleeping potion so the cat would fall asleep. The player can then stealthily sneak past the cat and open up new levels on the other side.

3.4 Sound design

The sound design for this project was very simple, as that was not our focus. We added a few sounds to just make the game a little more immersive and enjoyable. There are two main components of the sound: the sound effects and the background music. The background music (2) was chosen because it sounded nice and fit with our game theme. There are many different genres of video game background music. We had to go through a lot of songs to choose the one we though best fit our game. The song chosen has "Fairy" in its name, which we thought fit with the whimsical and magical theme of our game. Our animations and scene design are meant to look cute and bright, so it was only fitting that the music fell into the same category. The sound effects (3) are found on a royalty free site which had over 2,000 different sound effects. When choosing the right sound, we simply searched for the object name to see what options we could select from. But often, the sounds we chose were not meant for the object or animation we applied them to. The buzzing sound that the fireflies make is from a sound effect called "Wasp", and the digging sound in the cabbage patch is from a sound called "Dig in a Cat Litter Box." When listening to these sounds attached to a firefly or a cabbage patch, the player won't think that the sounds are out of place, so that is why this sound design works.

The sounds are also meant to help the user play our game. For example, a few players struggled to notice when they picked up a recipe item. We changed the UI a bit, but also added a sound to alert the player. The sound gives the player an extra alert that they are closer to finishing the level. Also, a door creeking noise was

added to the shed when the fireflies are all added. This is to alert the player that the firefly task triggered the door to open and they can now go through.

3.5 Inventory design

The inventory is the way that the player sees and uses its held items, which are treated separately from any ingredients in the level. It is controlled mainly by one InventoryController, which tracks the inventory, combines items when necessary, tells other scripts what items are available, and displays the items inside the game.

All items that can be directly picked up use the HoldCollectible script. This script has variables unique to each item that describe what items are needed in the inventory to pick up the item, as well as what item, if any, needs to be equipped. When the player tries to pick up an item, these prerequesites are checked, and if they are met the HoldCollectible script tells the inventory that the item was picked up, passing in the item and its name to the inventory.

Here, the InventoryController takes over. It stores the inventory as a linked list containing the actual GameObjects of each item, as well as a string to be a more simple name. It provides functions for adding items, removing items, and checking both if an item is in the inventory and if it is equipped. It also checks during its update if any items can be combined (for example, two individual fireflies into one item with both of them, or combining the stick and yarn to make a net).

In terms of UI, it manages two displays. The first is a constant display on the bottom-right of the screen showing the currently equipped item. When the player presses z, this item is switched to the next item in the linked list. This item is then moved to be held by the mouse on the screen, and its sprite is displayed on the equipped panel.

The second thing it manages is an inventory display which can be toggled on by holding the i key. This display uses a panel and two image objects (because there are always at most two items in the inventory in this level - this is scalable per level by using an array of image objects) to show which items are currently in the inventory. This works by looping through the inventory, starting from the current item, and adding each item to the inventory display sequentially. WHen the player releases the i key, the inventory disappears.

3.6 UI design

This section briefly describes important concepts of building User interface in Unity which were used in the implementation of the prototype.

Canvas

UI in Unity is build on Canvas (4), which is an area, which contains all parts of UI and controls how they are displayed. It is created as a Game object with Canvas component and UI elements must be its children.
In the prototype, Screen Space display render mode is used. This allows building responsive UI which is displayed on top of screen and is independent of camera. The size and position of elements changes to match screen size and resolution.

Layout design

Rect Transform is a transformation component specifically designed for UI, which includes a layout concept called Anchors. This allows to anchor the position of child elements respective to their parent. For example Potion Recipe UI (Section 3.6.1) is anchored to the top left corner of the Canvas, and stays in there regardless of screen size. In the Figure 3.7 there is a scene view of the Canvas and the Recipe UI. The four arrows signify the position of the anchor.

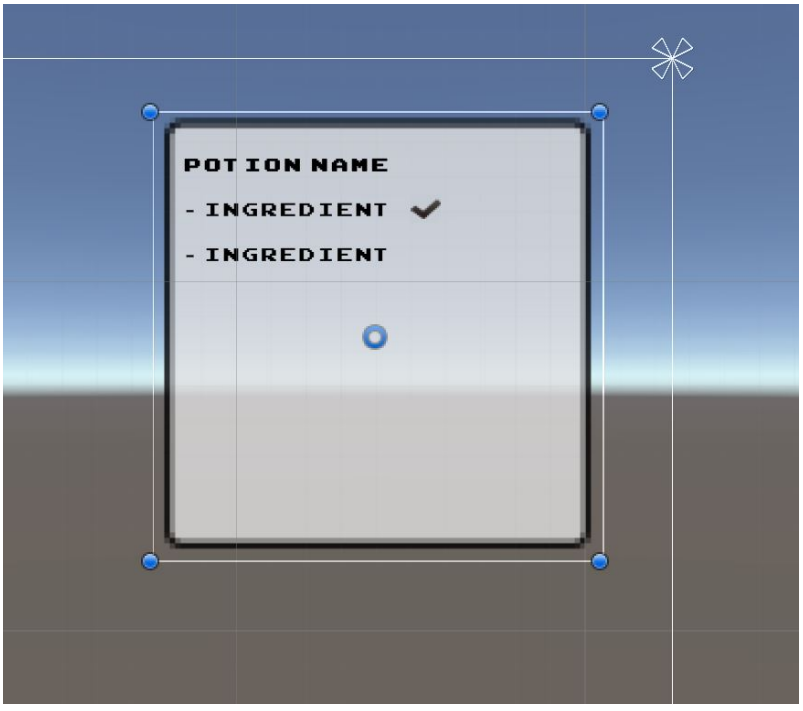


Figure 3.7: Anchoring of UI Element

Text Mesh Pro

The key package used for rendering text in Unity is Text Mesh Pro (5). Compared to the default Text functionality, TMP package has a better performance and more options in customizing the text appearance. It allows to usage of HTML and inserting sprites in the text.

3.6.1 Potion Recipe and Ingredient Collection

The goal of the game level is to collect all ingredients to create a potion. To achieve that several functionalities has been implemented.

- Recipe UI which displays the list of ingredients and marks which has already been picked up.
- Recipe Manager object which manages the progress of the collection process.
- Item Collectible script which handles the ingredient pick up events.

Potion Recipe UI element (Figure 3.8) is used to display which ingredients need to be collected to create potion in current level. When one of the required ingredients is acquired, a mark appears next to the ingredient name in the recipe.

Figure 3.8: Potion Recipe UI

The Recipe UI consists of a rectangle panel, which displays a background and Text Mesh Pro object which displays the recipe. Using TMP allows us to generate

checklist of potion ingredients which uses HTML tags for formatting and inserts sprites next to the ingredient name to signify that the item has been collected.

Ingredient collection scripting

The Recipe UI uses `UIRecipeController` script which changes the text displayed in TMP text field using `ChangeText()` method.

The scene contains empty Game Object called `RecipeManager` which holds script of the same name. The script several functions:

- Maintaining a list of ingredients for current level potion and the information whether they have been collected or not.
- Controlling the text displayed in the Recipe UI.
- Handling ingredient collection event.
- Triggering the level ending.

On the game start, the potion name and recipe are loaded and passed to `SetRecipe()` method, which creates dictionary representing the ingredients and their collection status. Afterwards `IngredientsToText()` method is called to generate rich text formatted by HTML tags.
`RecipeManager` contains reference to UI game object. To display the recipe the generated text is passed to the `UIRecipeController` script which manages the Recipe UI.

When an ingredient is picked up `itemCollected()` method is called from the trigger. In case the item part of the recipe, the ingredients dictionary is updated with the information that the item has been collected. Afterwards the recipe text is generated again using `IngredientsToText()`, and displayed in the UI with a mark next to the newly obtained ingredient and notification sound is played. The script also checks how many ingredients has been acquired. When the last one is collected, a message is displayed using Dialogue system (Section 3.6.2) to indicate that the level is finished and the mouse should head back home.

The ingredient Game Objects use Box Colliders 2D which serve as trigger when the mouse collides with the item. The `IngredientCollectible` script handles trigger events. It contains a reference to `RecipeManager` Game Object which allows to call its `itemCollected()` method passing the ingredient name, so the Recipe Manager can handle the collection process.

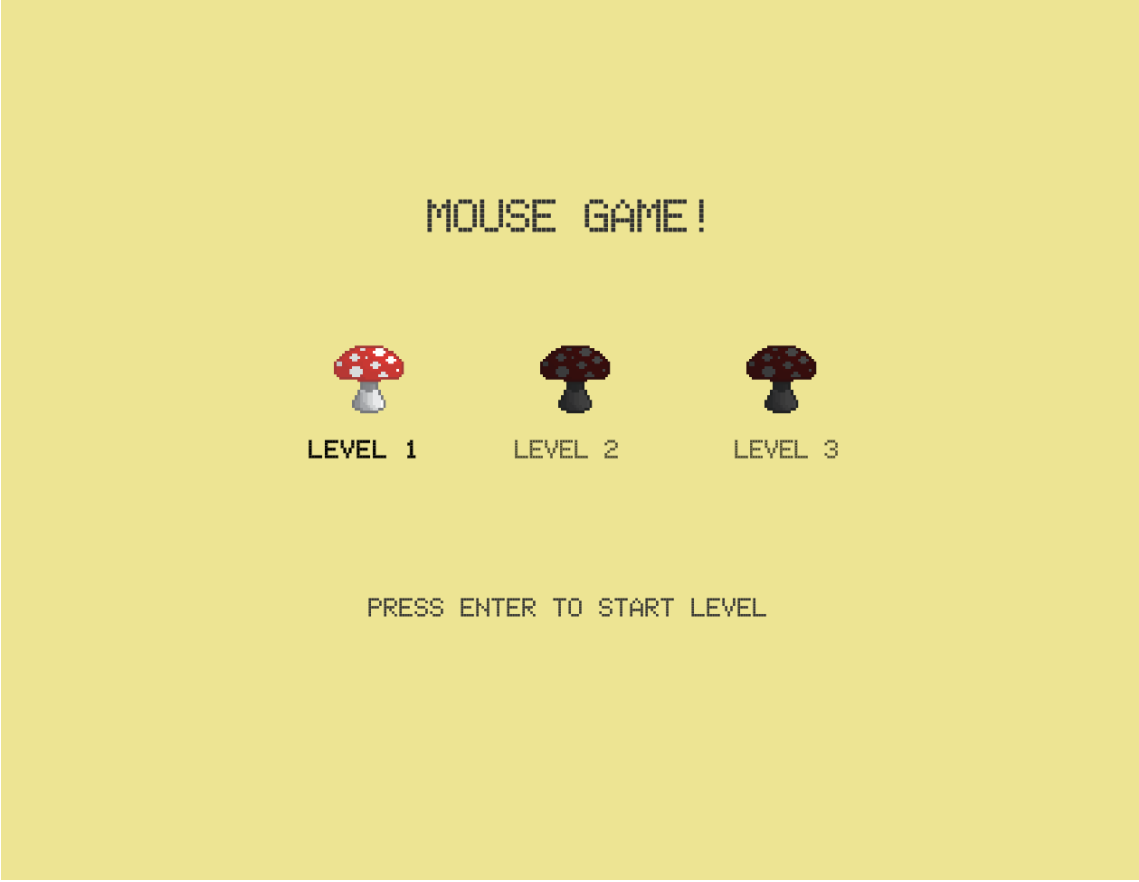
3.6.2 Dialogue System

The dialogue system is made very generically. It can be bound to any asset, so a dialogue can start at interaction. It can also be triggered after time or with the position of the character. We chose to make it generic so as the development of the game progresses, the dialogue system can handle all the different usages.
The system consists out of three main elements: the Dialogue box, the Dialogue manager and the Dialogue trigger. The dialogue box is just a canvas containing a text field and a button to continue. The text field is manipulated by the dialogue manager. The continue button triggers the dialogue manager and you can also press enter to trigger this.
The dialogue manager has a start method, which binds the dialogue box to the script, and makes it inactive (so it will not immediately appear. It also initialises the sentences as a queue which can be added later when the dialogue is triggered. The StartDialogue method takes a dialogue object as parameter. The dialogue object contains a string array for the sentences the dialogue should display and a string for a name. The name string is not used as of now, but was mainly made for future interaction with NPC's, so the names of NPC's could be easily displayed. The StartDialogue method sets the box to active so it is visible and then puts all the sentences from the dialogue object in the queue. It then calls DisplayNextSentence. DisplayNextSentence checks whether the last sentences has been displayed and in that case terminates the dialogue using EndDialogue. It otherwise dequeues the current sentences and displays is. The EndDialogue method just sets the box to inactive.
The dialogue trigger can be bound to objects, which calls the StartDialogue method at the trigger. We did this in the game using a collider for items (to explain how to

interact with items).
To make message display easy for the other developers to use, we added message display method. The purpose of this is to easily display a single sentences anywhere in the game. It is used when all the items have been found and it is time to go home. It just takes a sentence as a parameter and inputs it in the previously explained methods.
The dialogue system was made, partly using a tutorial. (6) The Arcade Classic Font was used for the text display. (7)

3.6.3 Level Select

The level select scene is a very simple but important scene. It is meant to be the first thing the player sees when starting up the game. The level select scene had a few versions before the final version was created. The first version was simple and only there as a place holder. When clicking on a level that we had not yet created (Level 2 or 3), a message was printed to the console. This level select scree did it's purpose, but was not so much in the aesthetic of the game.



When polishing off the game, we thought it would be easy and nice to make the level select screen a little nicer. The simple additions of characters and adding

a background make it feel like a real level select screen. The fireflies all use a script, Flying.cs, in order to fly around in random directions. The randomness is constrained to a box around the firefly's start position, so that they don't fly over the text and cause too much of a distraction. The level select feature also brings up a pop up box if the player tries to select a level that we have not yet developed. If more levels were added, this level select screen could easily accommodate more options.

* Ingredients system

3.7 Implementation issues/difficulties

There were many small issues, as most of us were new to Unity at the start of this project. Luckily, most issues were either able to be solved or a new solution would be met. For example, the shed was initially set up as a new scene. But after trying for hours, we couldn't figure out how to carry objects through scenes and also keep objects in previous scenes the same when the player returned. To fix this, the shed was appended to the bottom of the map, meaning the player is teleported to a new location rather than a new scene. This is probably not the best way to do it, but the shed still gives the idea that the player changed environments entirely even though it was simply right below.

4 Analysis

4.1 Playability

The playablility of our game is very straightforward. With simple controls, a clear objective, and an endgame in sight, the player should not have any issues picking

this game up and enjoying it. Our game is an adventure and puzzle game combined, which creates a mix of fun and challenging. Because of our novice skills in Unity, some issues are still in the game that impact playability. It can be somewhat difficult to pick objects up just based on the fickleness of our implementation. With more time and experience, those issues could be fixed.

4.2 Feedback

We tested the game in class with our peers as well as with some friends. We got the following feedback, either specifically from the players or as things we noticed while they were playing.

- It is not clear which items you have while playing. There should be some sort of display to see all the items that are currently held.
  - We implemented an inventory screen. By pressing 'i', the player can now see all their held items in the inventory.
- The recipe book (canvas in the upper left corner) is not clear enough. Since the goal of the level is to get all these items, it is hard to notice it at the start and therefore hard to know the objective. The recipe book should be more noticeable.
  - To make the recipe book more noticeable, we made it slightly bigger, less see through and used a different font.
- The mouse is too slow.
  - We made the mouse move slightly faster.
- It is annoying that you have to use the mouse to click away the dialogue messages, since all other controls are on the keyboard. Clicking 'continue' on the dialogue should be possible through the keyboard as well.
  - We implemented a feature to also be able to press enter to continue on dialogue.
- It is hard to see whether you actually picked up an ingredient. The display only shows a small mark when an ingredient is picked up. This is especially troublesome when harvesting cabbage since nothing is removed at picking it up. There should be some feedback to the user when at item is picked up.
  - Along with the item appearing in the dialog box, a congratulating sound is also played to alert the player.
- The cat is too fast. It catches the player too quickly when it tries to explore the cat's living space. We saw many people who consequentially either thought nothing was there or who tried to find a way to kill the cat.

  - We made the line from where the cat starts running toward the player a little further. This way the cat will not catch the player until he is really too far.

- There should be an overall map of the level so the player has more overview.
  - We added a simple tool where the user can press the letter "m" on the keyboard and an overall view of the area is shown. This was very easy to implement and such a good idea from the feedback, so we were able to do this very fast.
- It is unclear that the cabbage should be found at the patch at the right upper corner. This should be made more clear.
  - We placed a sign with a cabbage, making it much more clear.
- It is unintuitive that red yarn and a stick make a net.
  - The yarn is now white. It should be more clear that a stick and white yarn can make a net.
- The story of the level is unclear, which makes the objective unclear. Make sure there is an explanation of the objective.
  - A cut scene at the start of level has been implemented. It explains the story of the mouse and the objective of creating a potion.

5 Conclusion

5.1 Game evolution

The idea for this game started with the story. We wanted to tell the story of a mouse that had to explore a garden in order to turn back into a human. As we started developing the game, the story stayed the same, but our ideas for puzzles and the aesthetic of the game changed. The first version of the game included a mouse that didn't look like it fit in, only one puzzle, and a very under-furnished map. As we continued to develop it, we were able to bring more to life, like the cat chasing the player and a second area (the shed).

We also thought we would have time to create multiple levels with even more puzzles, but we were not able to do that as we focused a lot on the first level and making it complete.

5.2 What would you do differently?

If we would start making the game again, we would make a more clear plan from the start. We started with the concept, but made up most of the levels and mechanics along the way. The puzzle and level were not clear from the start. It could have

saved us time if there was a comprehensible plan. All in all it did turn out well, but this could have saved us some time and possibly made time to implement more features.

5.3 Next steps

- Implement more levels: there are a number of ways to open up new levels. We could make the player be able to pass by the cat and open up more ground from there; we could make the player able to open the shed and do a puzzle in there; we could use our title screen to go to a whole new level. In these levels the player could make even more potions, collect items and bonus stars.
- Graphics: the graphics so far are rather minimum. In next steps we could make the mouse move in a better way by using multiple sprites for directional walking. We could make more animations for objects and potion collection. In addition, if we were to create our own graphics we could fit more clues to the puzzles within the graphics. For example, we could make part of the fence look weaker so that players know they can break it somehow.
- Further develop the storyline and main flow of the game: Right now, we have a basic idea for the story, but we have not outlined how it would develop level-by-level. Similarly, as our focus was on making one level as an example, we have not thought much about how each level fits into the overall progression, and when we want to open new areas of the map. In the future we would need to work on combining these elements of story and gameplay to develop a cohesive plan for how the goals and levels progress.

References

[1] ChaoticCherryCake, "Pokemon tileset from public tiles 32x32," DeviantArt. [Online]. Available: https://www.deviantart.com/chaoticcherrycake/art/Pokemon-Tileset-From-Public-Tiles-32x32-623246343

[2] LucidGate, "Skyward sword music: Fairy theme (fan made)," Youtube. [Online]. Available: https://www.youtube.com/watch?v=30SVnH2Gt4A

[3] SoundBible, "Sound bible," SoundBible. [Online]. Available: https://www.soundbible.com/

[4] U. Technologies, "Unity manual: Canvas," Unity Technologies. [Online]. Available: https://docs.unity3d.com/Manual/UICanvas.html

[5] D. N. Studios, "Text mesh pro documentation," http://digitalnativestudios.com/textmeshpro/docs/, accessed: 09.12.2019.

[6] Brackeys. How to make a dialogue system in unity. Youtube. [Online]. Available: https://www.youtube.com/watch?v=_nRzoTzeyxU

[7] J. Fischer, "Arcade classic font," https://www.1001fonts.com/arcadeclassic-font.html, accessed: 09.12.2019.