The Last Cookie

By Funmi Falegan, Karan Kaur, Ahartisha Selakanabarajah, Anna Robertson and Jo Baker

1. Introduction

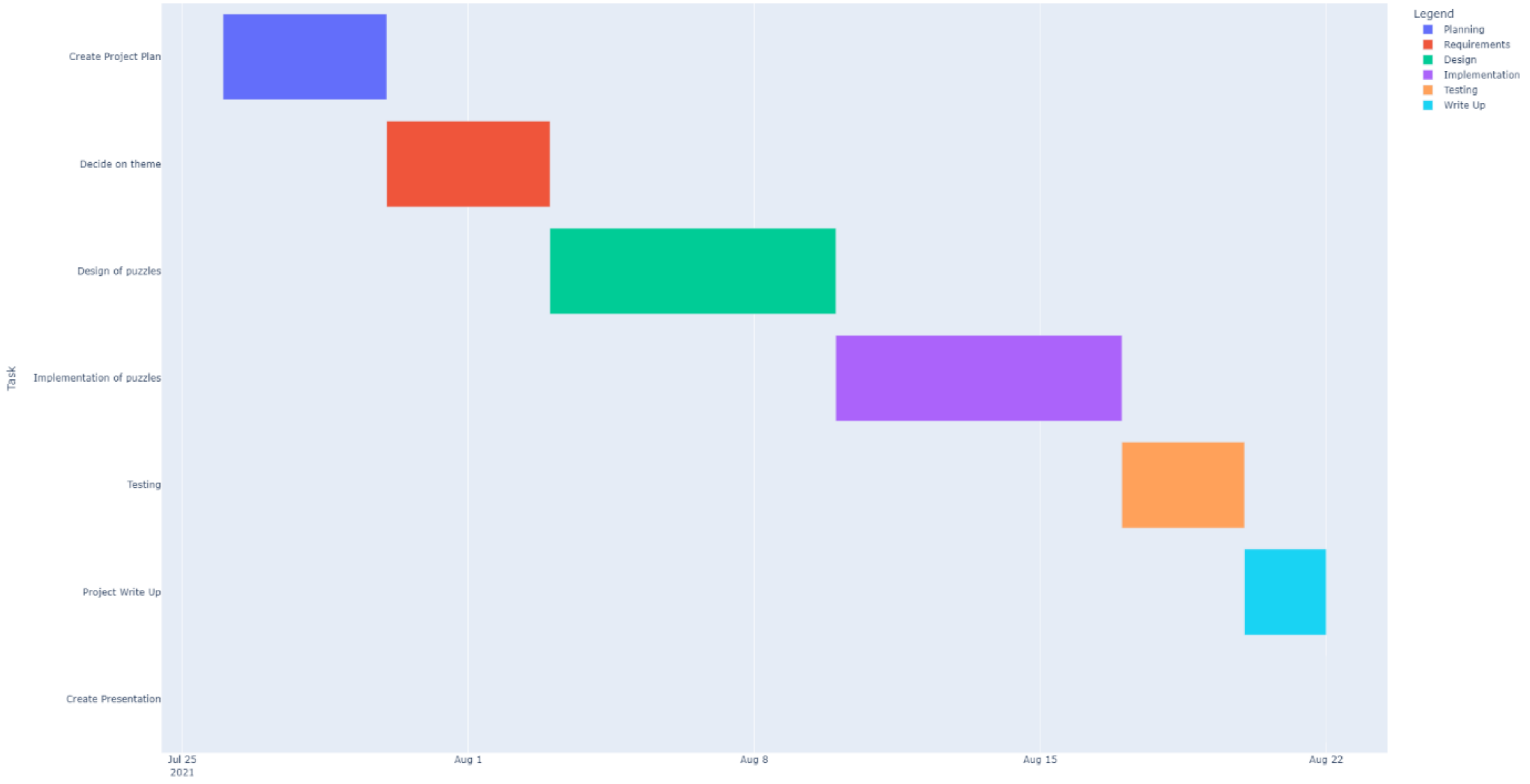
1.1 Aims and Objectives

The aim of this project was to design a text based adventure game with graphical puzzles using a combination of object-oriented programming (OOP) and implementation of modules such as turtle, pygame and arcade.

Our objectives included:

- Coming up with a theme and name for the game
- Making a main file that runs the game
- Designing a map of the game (layout of rooms)
- Creating a Room class, this would include:
    - Room name
    - Room description
    - Other rooms linked to that room
- Creating 5 puzzles, one for each room
- Writing a storyline

1.2 Roadmap


Task
Create Project Plan
Decide on theme
Design of puzzles
Implementation of puzzles
Testing
Project Write Up
Create Presentation
Jul 25
2021
Aug 1
Aug 8
Aug 15
Aug 22
Legend
Planning
Requirements
Design
Implementation
Testing
Write Up

2. Background

We decided to name our game 'The Last Cookie' and theme it around a detective looking for clues to try and find out who ate 'The Last Cookie'. This would include navigating the 'Gingerbread House' through a text based system and collecting clues by playing and winning puzzle games.

Our game blurb is as follows:

> "There is a shortage of cookies in Biscuitland™ and someone has just eaten the last one! You are a trainee Digestive Detective™ and you have been tasked by Sherlock Scones ™ to discover who ate The Last Cookie™. Visit the rooms in the Gingerbread House solving puzzles to unlock clues so you can report the identity of the Cookie-culprit to Sherlock Scones! ™. Your reward? Plenty of dough... let's see how the cookie crumbles..."

2.1 The Rules

- The detective player must complete a puzzle to earn a clue.
- The puzzles will consist of minigames: a platform game, hangman, pong, connect four and a maze.
- The detective player must earn 5/5 clues to find the suspect and complete the game.

2.2 Logic

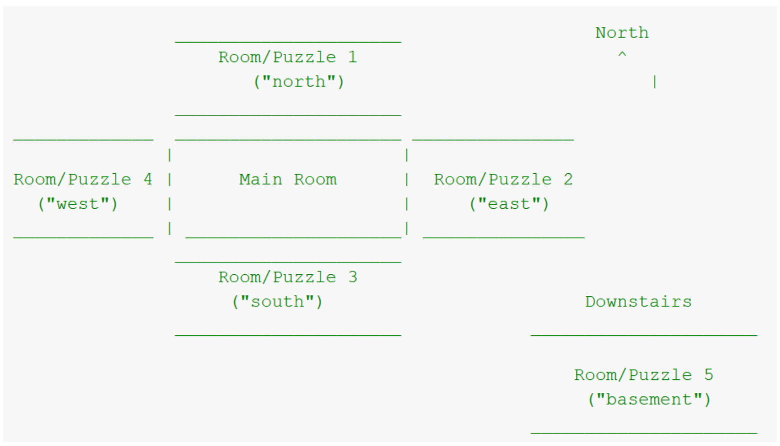


Figure 1: A basic layout of the gingerbread house for navigation

- The detective player will navigate the gingerbread house via a text based system. They will be asked which direction they would like to go and given a description of each room available. The layout of the rooms will have a central room with 4 rooms coming off north, south, east, and west and a 5th room in the basement (**figure 1**).

- The puzzle games will implement a variety of controls for movement such as ‘WASD’, arrow keys and mouse as well as special functions through other keys.
- There will be an inventory for collecting the clues that will be updated when a player successfully completes their puzzle game. When the player has 5 unique puzzle clues in their inventory, they will be able to complete the game by speaking with Sherlock Scones in the main room of the house.
- When a player successfully completes a puzzle that room will be locked to signify that it is complete.
- We decided that there would be no way to lose the game (no health/lives etc), rather, the player would have to repeat the puzzles until they successfully gained all five of the clues and won the game.

3. Specification and Design

3.1 Functional Requirements

The following requirements are to achieve the primary objectives of the project:

- The game will have a title screen with buttons to start or exit the game
- The game will have the option to quit out the game during play
- The game will have an option to call a map of the Gingerbread House
- The game will have an option to call an instructions screen
- The game will register when a puzzle has been completed and update the play inventory
- The game will have an option to call the inventory to check which clues have been obtained
- If time allows a save function will be implemented

3.2 Non-Functional Requirements

- The code will be structured and maintainable
- The code will be well laid out with appropriate comments so that documentation can be automatically generated

3.3 Design

The game will give instructions to the player at the start of the game to tell them how to play and what they need to do to win so they have a focal point and understand what they are meant to do. Throughout the game the player will be able to return to the central room to speak to Sherlock Scones or check their inventory to find out which rooms/puzzles they still need to complete, and which clues they have already obtained.

The player will be in control of the order that they complete the puzzles, and the room descriptions will give a hint to the type of puzzle contained in the room to help them make a choice on a puzzle they will enjoy playing at that time.

The pacing of the game will be at the speed that the player choses. If time allows a save feature will be implemented so that the player can come back at a time suited to them to continue playing without having to restart completely.

3.4 Architecture

The main game will be built simply using OOP in an integrated development environment utilising Python 3. Each of the puzzles will implement different modules available in Python 3 such as:

Platform Game:

- Arcade: this is a module used to create 2D video games.
- Pygame: this is a cross-platform library used to develop multimedia applications such as video games

Hangman:

- Pygame: this is a cross-platform library used to develop multimedia applications such as video games
- Random: this module is used to generate random numbers.

Pong:

- Turtle: this module uses tkinter for its underlying graphics and provides turtle graphics (such as creation of pictures and shapes) in both object-oriented and procedure-oriented ways on a virtual canvas.

- Winsound: this module provides access to the basic sound-playing machinery provided by the Windows platform.
- Random: this module is used to generate random numbers.

Connect Four:

- Pygame: this is a cross-platform library used to develop multimedia applications such as video games
- Random: this module is used to generate random numbers.

Maze:

- Turtle: this module uses tkinter for its underlying graphics and provides turtle graphics (such as creation of pictures and shapes) in both object-oriented and procedure-oriented ways on a virtual canvas.
- Time: this module provides various time related functions.

4. Implementation and Execution

4.1 Development Approach

Our team worked using the Waterfall approach. Due to time constraints many parts of the puzzle needed to be worked on at once. We created a project plan prior to starting this project which we followed as a guide. We decided on the requirements for the game and the theme it would take. Next, we each designed and implemented a puzzle game for each room that could be called from the main text based part of the game. Once this was completed, we created the Room class and the main game file and created a storyline that executed the puzzles once the player entered the corresponding room. Finally, tests were written for the main game file and each of the puzzles.

4.2 Team Member Roles

Funmi Falegan:

- Creation of the Connect Four game

Karan Kaur:

- Creation of the Hangman (Save Gingy) game

Ahartisha Selakanabarajah:

- Creation of the Platform (El Do-Cookie-Rado) game

Anna Robertson:

- Creation of the Pong (Baguette Tennis) game
- Creation of the game blurb

Jo Baker:

- Creation of the Maze (Maryland Maze) game
- Creation of the main text-based adventure file

Whole team:

- Planning of the project
- Theme of the game
- Creation of characters
- Creation of storyline
- Implementation of testing

4.3 Tools and libraries

Each of the puzzles will implement different modules available in Python 3 such as:

Platform Game:

- Arcade: this is a module used to create 2D video games.
- Pygame: this is a cross-platform library used to develop multimedia applications such as video games.
- Tiled: this is a free, open source 2D game level editor that produces files that can be read and used by the arcade library.

Hangman:

- Pygame: this is a cross-platform library used to develop multimedia applications such as video games.
- Random: this module is used to generate random numbers.

Pong:

- Turtle: this module uses tkinter for its underlying graphics and provides turtle graphics (such as creation of pictures and shapes) in both object-oriented and procedure-oriented ways on a virtual canvas.
- Winsound: this module provides access to the basic sound-playing machinery provided by the Windows platform.
- Random: this module is used to generate random numbers.

Connect Four:

- Pygame: this is a cross-platform library used to develop multimedia applications such as video games.
- Random: this module is used to generate random numbers.

Maze:

- Turtle: this module uses tkinter for its underlying graphics and provides turtle graphics (such as creation of pictures and shapes) in both object-oriented and procedure-oriented ways on a virtual canvas.
- Time: this module provides various time related functions.

4.4 Implementation process

4.4.1 Achievements

Given the circumstances under which we made this project – having a game is definitely an achievement! We worked and developed extremely well as a team and adapted well to unforeseen circumstances such as illness. Further to this, each member had personal achievements through each of their individual roles in the projects:

Funmi Falegan:

- Learning how to utilise the pygame module.
- Creating graphics and scaling them to suit the aspect ratio of the game
- Gaining a better understanding of game concepts such as physics and DeltaTime
- Adding music and sound effects
- Incorporating object-oriented programming

Karan Kaur:

- Gaining a better understanding of pygame and how to implement graphics into an otherwise simple program.

Ahartisha Selakanabarajah:

- Learning how to utilise the arcade and pygame modules.
- Manipulating graphics to suit the aesthetic of the game using Tiled.
- Getting a better understanding of game physics.
- Implementing collision detection when landing on platforms and reaching the goal.
- Adding music and sound effects.
- Incorporating object-oriented programming.

Anna Robertson:

- Learning how to utilise the turtle module.
- Designing graphics.
- Getting sound to work.
- Getting balls and bats to move as well as implementing collision detection.
- Creating a simple AI for the computer player.
- Ball physics.

Jo Baker:

- Learning how to utilise the turtle module.
- Creating graphics to place in the maze that were the correct size.
- Learning to utilise the program GIMP to create graphics.
- Initialising collision detection on maze walls.
- Implementing coordinates for picking up items within the maze.
- Creating the option to quit or replay the game after the game was won.

4.4.2 Challenges

As a team this project was very challenging. Our team suffered from some personal life events which meant they were unavailable to attend meetings or work on their projects. Our team adapted to this to allow the team members time to address these events, however, given the time constraints on this project, any loss of time sets back the project by quite a few steps.

It was also difficult to find times for meetings where the whole team could make it, this often meant we were having meetings with only 3-4 of the team and we had to update the missing members via our slack channel.

Furthermore, due to other commitments team members have/had such as work or study this further limited available time.

Each team member also encountered individual challenges during the project:

Funmi Falegan:

- Learning pygame: learning about game physics and state machines.
- Issues with co-ordinates when detecting a winner
- Using maths concepts such as co-ordinates and ratios when scaling and rendering graphics onto the screen
- Fixing the music so it plays clearly across the game loop
- Learning about large scale project structures and naming conventions
- Struggles with loading images, had to utilise os.path.join()

Karan Kaur:

- Ensuring different conditions were met e.g. changing the splash screens and exits depending on whether the game was won.
- Getting used to pygame and refactoring the code so OOP was used.

Ahartisha Selakanabarajah:

- Editing of sprites and tiles so that they had the same dimensions which was time consuming.
- Getting the end screen to work after the final level.
- The movement of the player was awkward when moving up or down ladders.
- Implementing a scrolling feature wherein the program focuses on the movement of the player.

- Attempting to add extra features in the time given e.g. moving platforms, enemies and a scoreboard.

Anna Robertson:

- Keeping code as simple as possible.
- Trying to get the ball to move and increase speed properly.
- AI was initially impossible to beat – had to change the bat speed.
- Getting the splash and end screens to work.
- Game speed may play differently on different computers.
- Sound only works on Windows machines.
- Moving the game from a simple game to OOP with classes and methods.

Jo Baker:

- Moving the game from a simple game to OOP with classes and methods.
- Creating .exe files
- Pulling all 5 mini games into the text based adventure and getting the games to run when the player entered the corresponding room

4.4.3 Changes

We had originally planned our project to make a purely text based game, but quite quickly decided we wanted a harder challenge and decided to implement the puzzles as individual mini games using a module of our own choosing. This enabled us to be as creative as we wanted, whilst also allowing each of us to dedicate as much or as little time to the game that our personal circumstances permitted.

5. Testing and Evaluation

5.1 Testing Strategy and Functional Testing

Our testing strategy included mocking inputs of functions such as distance from specific co-ordinates within the game (such as picking up an item or crossing a winning point) to see if game states changed or a Boolean value was updated. For this both pytest and unittest were used; mocking was achieved through the use of the decorator @patch(‘function_to_mock) above the test, whilst also passing in the values to the test function, for e.g.:

```
@patch('maze_puzzle.Game.is_detective_near_winning_position')
@patch('maze_puzzle.Game.true_cookie_crumbs')
@patch('maze_puzzle.Game.false_no_win')
@pytest.mark.me
def test_win_is_false(mockis_detective_near_winning_position,
mocktrue_cookie_crumbs, mockfalse_no_win):
    win = maze_puzzle.Game()
    win.game_state = "game"
    win.init_game_state()
    win.true_no_win()
    mockis_detective_near_winning_position.return_value = True
    mocktrue_cookie_crumbs.return_value = False
    mockfalse_no_win.return_value = False
    win.win()
    assert win.no_win is True
```

5.2 Limitations

Unfortunately, the testability of pygame proved challenging as game logic is often not suited to unit tests. If time had allowed, integration testing could have been explored and if we were creating a game that was going to market, a company that employs automated quality assurance could have been utilised. However, due to these limitations no tests could be conceived for the mini games written using this module.

6. Conclusion

We created a game that is a combination of a text adventure alongside mini puzzle games including: a platform game, connect four, hangman, baguette tennis and a maze.

Each time the player enters a room, a graphical puzzle game loads and the player earns a clue. With five clues collected the player can complete the game. However, with more time we would have liked to have made it so the clues were only awarded upon completion of the puzzle in each of the rooms. We were also unable to get puzzle 4 working in our format as the arcade module seems to have compatibility issues with packaging to a .exe file. Furthermore, we would like to have implemented the whole game as a graphical world with interactive character sprites giving a scripted story to the player; this would include storylines for the clues given from each puzzle, allowing the player to work out who the suspect was through a process of elimination.

Our team worked extremely well together and adapted to the needs of each individual as required. This included modification of our project plan every few days based on the realistic timescales each of our tasks was taking.

In conclusion, we accomplished many of the goals of our project and delivered five individually working puzzles alongside the main game, in which four of the puzzles were implemented and that are fun to play.