Technological Basics
Lilli Heidinger & Maja Menzel

29.06.2025

Documentation - Final Project

Group work: “Cat Simulator”

Our game is a 2D point-and-click/side-scrolling adventure built with Pygame. The player controls a cat exploring its owner’s apartment in search of food but quickly finds the culprit behind its empty food bowl. The game combines exploration, an inventory system, and multiple endings (win fight, lose, or secret love ending) depending on item interactions.

Development Timeline:

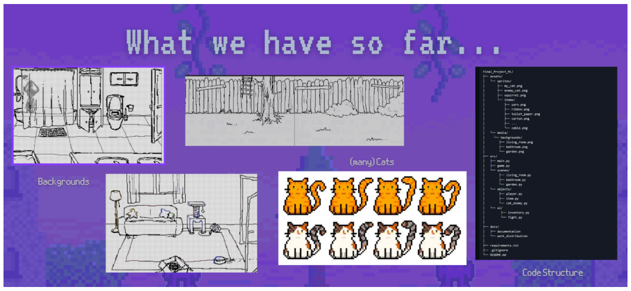
29.06.2025
Lilli & Maja:
- a (very chaotic) Mural mindmap:
    - first group task distribution & To-Do List

01.07.2025
Lilli:
- create a new repository and clone it to local machines
- configure .gitignore file and README.md

Maja:
- work on sprite sheets for player
- work on intermediate presentation

06.07.2025
Lilli:
- add code skeleton to Github repository
  → used AI to outline what files would be needed (see attachment M1)

08.07.2025:
- intermediate presentation



What we have so far...

Backgrounds

(many) Cats

```
Final_Project_ML/
├── assets/
│   └── sprites/
│       ├── my_cat.png
│       ├── enemy_cat.png
│       ├── squirrel.png
│       └── items/
│           ├── yarn.png
│           ├── ribbon.png
│           ├── toilet_paper.png
│           ├── carton.png
│           ├── ...
│           └── cable.png
│   └── media/
│     └── backgrounds/
│        ├── living_room.png
│        ├── bathroom.png
│        └── garden.png
├── src/
│   ├── main.py
│   ├── game.py
│   └── scenes/
│       ├── living_room.py
│       ├── bathroom.py
│       └── garden.py
│   └── objects/
│       ├── player.py
│       ├── item.py
│       └── cat_enemy.py
│   └── ui/
│        ├── inventory.py
│        └── fight.py
│
├── docs/
│   ├── documentation
│   └── work_distribution
│
├── requirements.txt
├── .gitignore
└── README.md
```

Code Structure

06.07.2025
Lilli:
- finish background paintings for scenes and upload to the repository

21.07.2025
Lilli:

- sketch game logic: “state machine”
- implement menu/start screen → use pygame rects to create buttons that lead to start/end of the actual game

Maja:

- work on walking sprite art

22.07.2025
Maja:

- finish sprite sheets and walking/idle cat animation

Lilli:

- implement living room (with working cat animation/player controls) → cat sprite sheet is tied to player key-inputs and moves on a horizontal line above the background image

see how the game looks from here on out:
https://github.com/foreverearthmover/Final_Project_ML/blob/main/docs/documentation.md

23.07.2025
Maja:

- creating an item inventory system with picking up and drawing, also hover messages
  → rect-mouse collision with all items by placing separate pngs of items above the background, hovering displays specific messages for items (ex. Cat tree: "I could sharpen my claws on this.")
  → clicking items adds them to an invisible inventory (max. 4 items)

24.07.2025
Lilli:

- add bathroom.py and garden.py for scene transitions
  → similar to living_room.py
  → to do: add scene switching method to game class and modify living_room.py to “lead” to other rooms

Maja:

- making an inventory appear and make it possible to drop items from the inventory
  → displayed by pressing “E”, shows the items picked up and the option “drop”

25.07.2025
Lilli:

- implement scene transitions in alternative way: buttons for navigation instead of walking off-screen
  → **preliminary solution** so we can work on other aspects of game
  → tried to make walking off-screen work but couldn't manage because hit-box of sprite sheet is inaccurate; checking cat's position against boundaries of screen doesn't work
- added a red rect for the squirrel and a blue rect for the neighbour cat (placeholder for actual sprite sheets)
- handle *squirrel chase animation* by adding action/chase button after triggering running away animation for squirrel → *transition to other half of garden, end boss*

Maja:

- fixing many problems from merging :,)
- drawing new cat skins
- adding a skin selection screen
  → before the start screen, ability to choose from multiple, different looking cats

26.07.2025
Maja:

- fixing even more problems
- setting hover message, inventory etc. to new positions and color

27.07.2025
Lilli:

- fix the inventory → wouldn't show up because a line of code was missing
- extend item/inventory logic to garden and bathroom, inventory is "global"
- ensure that items return to the room they were found in when dropped

29.07.2025
Maja:

- starting to work on a stats system, with a display in the corner
  → important for endings, checking if item effects are calculated correctly
- implementing fight endings with placeholder images
- implementing new stat and item: Love (for secret ending)

Lilli:

- fix graphical issues → stats display, boss cat appearance, chase button disappearing
- implement *transition from boss fight to ending* (provisional)

30.07.2025
Maja:

- fixing bug in stats not actually being calculated
- adding a “used message”
- making stats disappear again once the item gets dropped
- resizing inventory and all messages

Lilli:

- **fixed hitbox problem** → root cause why position-based scene transitions didn't work (collision rect was as wide as the entire sprite sheet when it should’ve been a fraction of that width)
- revert scene management back to being based on cat position (draw_navigation_buttons commented out)

31.07.2025
Maja:

- trying to add screen bounds to prevent cat from walking infinitely off-screen and failing due to circular problems
- drawing more sprite sheets

01.08.2025
Maja:

- adjusting the bow item to the bow sprite sheet
- fixing the love ending

03.08.2025
Maja:

- fixing the bow sprite sheet so that it disappears again after dropping the item
- fixing all inventories, because there were two
- fixing the endings
- experimenting with stats

04.08.2025
Maja:

- implementing hidden object logic → game secret: clicking the cabinet in the bathroom “drops” 4 rolls of toilet paper but if you keep clicking a bow appears (hidden item) which allows you to entice the neighbour cat (love ending)

06.08.2025:
Lilli:

- complete bathroom item handling (hidden object logic) → after clicking the cabinet 3 times the bow appears and can be equipped
- prevent stats (damage, health, love) from being negative values and assign bow with love stat correctly (help from AI: see attachment M2)
  → **stats for ending complete**
- change font and background color for character selection

07.08.2025:
Lilli:

- finish drawing and insert proper ending screens
- “animate” *endings* and add *hissing sound* effect

08.08.2025:
Lilli:

- centralize fonts in assets\media\fonts.py
- make graphical improvements

09.08.2025
Lilli:

- add invisible wall to bathroom and garden → cat can’t go out of bounds

10.08.2025
Maja:

- finishing squirrel sprite drawing

11.08.2025
Maja:

- adding proper **squirrel** and bow pngs into the game
- adding squirrel idle animation

Lilli:

- improve bathroom scene by adding a filled vs. empty cabinet transition
- move draw_inventory() and draw_hover_message() methods into helpers to make code more maintainable
- work on squirrel and chase transition
- small adjustments to inventory position, cabinet hover message & bow scale

12.08.2025:
Lilli:

- add *instructions screen* before the character selection
- fix squirrel click hitbox
- generate requirements.txt file and add to GitHub

Maja:

- add squirrel walk animation

13.08.2025:
Maja:

- fix hitbox in character selection
- add a *walking animation when walking over to the boss fight cat*

14.08.2025
Maja:

- finishing and polishing cat walking animation for the chase button

15.08.2025
Maja:

- final touches on making texts centered

18.08.2025
Lilli:

- clean up code and add comments

07.09.2025:
Lilli:

- last adjustments to the code: add some more error handling, asked JetBrains AI whether we missed any errors in the code
- add shower sounds upon entering the bathroom

14.09.2025:
Lilli:

- change relative to absolute paths to ensure running game from terminal works
- update README file

Technological Basics 29.06.2025
Lilli Heidinger & Maja Menzel

Work Distribution

Maja	Lilli
Sprite Sheet art & animation	Scene Management/Transitions
Player Class	Background & Item art
Items & Inventory	Git Repository
Stats based on Items	Menu and Instructions Screen
Character Selection	Boss Fight Scenes

References

Fonts

8 Bit Wonder Font | dafont.com → big font
Retro Gaming I dafont.com → small font

Sounds

Minecraft Cat hiss by MollyKate - Meme Sound Effect Button for Soundboard - Voicemod
https://mixkit.co/free-sound-effects/discover/shower-indoor-ambiance/

Tutorials

https://archive.is/eobVZ → first research into the idea of a point and click adventure
https://www.pygame.org/docs/ → Pygame documentation
https://www.youtube.com/watch?v=_FpkqIkeKUY → Hitboxes and Game Boundaries

Game State Management

▶ Pygame Menu System Tutorial Part 1: Game Loops and Structure
▶ HOW TO MAKE A MENU SCREEN IN PYGAME!
→ Menu System
▶ Pygame Game States Tutorial: Creating an In-game Menu using States
→ Game States
▶ Pygame Camera Tutorial: How to Make a Side-Scrolling Camera System
→ Side-Scrolling (Border Camera Method)

Technological Basics 29.06.2025
Lilli Heidinger & Maja Menzel

Sprite Sheets
How to do Pixel Art in Procreate - learning how to draw/export Pixel Art using Procreate
PyGame Beginner Tutorial in Python - Loading Spritesheets
PyGame Beginner Tutorial in Python - Sprite Animation
→ basics of Sprite sheet drawing and coding

Difficulties

- side-scrolling → scene transitions and sprite sheet interactions (hitboxes of player, background, items, UI, etc.)
- inventory → how to make sure items are dropped in the right rooms/don't duplicate & stats being removed after items being dropped which would cause negative stats
- implementing classes across files → often caused circular problems
- having one universal room class that each room inherits from → realized too late and didn't get it to work since we basically would have had to start over
- not accidentally creating conflicts on GitHub since we had to work on the same files

Future developments

- more rooms/endings → increase the scale of the game
- add cutscenes and more animations → smoother and more cinematic flow of the game, i.e., animations for the use of non-movable items
- possibly an interactive end scene with a fight system

Attachments

M1: ChatGPT Prompt for Code Structure

Prompt: “Please help me structure my GitHub repo for my game concept: In a 2 person group work, we plan a cat simulator game (1 dimensional) where a pixelated cat wakes up in the living room (drawn pixel art background, 300x200 px) with the objective of getting food. It can then decide to go left or right. To the right is the bathroom (also drawn pixel art background) and to the left the garden (also drawn pixel art background but 600x200 because when going further right the background scrolls with the cat). Throughout the rooms there are multiple items that have to be clicked using pygame object collision that either give an effect or are taken into the inventory to be used later, the cat is moved using WASD or Arrow keys. There are three endings: the first two come after a fight with the mean neighbour cat that stands at the very end of the garden (seen once you run far right), you can either lose or win the battle depending on the items you used or picked up. The secret third ending can be achieved by using an object from the living room in the bathroom and getting a ribbon. when facing the mean cat with the ribbon equipped, they will fall in love and the game ends.”

ChatGPT response:

GitHub Repository Structure

Your GitHub repo should be clean, modular, and professional. Here's a structure tailored to your cat simulator:

```
cat-simulator/
├── assets/
│   ├── backgrounds/
│   │   ├── living_room.png
│   │   ├── bathroom.png
│   │   └── garden.png
│   ├── sprites/
│   │   ├── cat_idle.png
│   │   ├── cat_walk1.png
│   │   ├── items/
│   │   │   ├── yarn.png
│   │   │   ├── ribbon.png
│   │   │   ├── fish.png
│   │   │   └── spray_bottle.png
├── docs/
│   └── documentation
├── src/
│   ├── main.py
│   ├── game.py
│   ├── scenes/
│   │   ├── living_room.py
│   │   ├── bathroom.py
│   │   └── garden.py
│   ├── objects/
│   │   ├── player.py
│   │   ├── item.py
│   │   └── cat_enemy.py
│   ├── ui/
│   │   ├── inventory.py
│   │   └── dialog.py
├── requirements.txt
├── README.md
└── .gitignore
```

M2: Help from AI with inventory

Prompt:

“Can you bring this code to work, items that are moveable should go into the inventory and apply effects to players stats. You should be able to drop them from there causing their effect to disappear and to them being back in their room. stats should not be negative if dropped”

```
# Inventory interactions
if self.game.show_inventory:
for i, item in enumerate(self.game.inventory):
item_x = INVENTORY_POSITION + 20 + i * ITEM_SPACING
item_rect = pygame.Rect(item_x, 20, 40, 40)

if item_rect.collidepoint(mouse_pos):
self.selected_inventory_item = item
return

drop_rect = pygame.Rect(item_x, DROPBUTTON_POS_Y, 50, 20)

if item == self.selected_inventory_item and
drop_rect.collidepoint(mouse_pos):

#Drop logic
item.picked_up = False
for item in item.picked_up = False:
item = dropped
#Undo stats if applicable (prevent negative values)
for i in room_item == dropped
current_stat = self.game.stats.get(item.stat, 0)
```

```
#Assign item back to its original room
for room_items in rooms.items():
if any(room_item["item"] == dropped):

target_room_items.append(item)
break

self.game.inventory.remove(item)
self.game.status_message = f"Dropped the {item.name}."
self.game.message_timer = pygame.time.get_ticks()
return
```

Jetbrains AI code:

```
# Inventory interactions
if self.game.show_inventory:
   for i, item in enumerate(self.game.inventory):
       item_x = INVENTORY_POSITION + 20 + i * ITEM_SPACING
       item_rect = pygame.Rect(item_x, 20, 40, 40)

       if item_rect.collidepoint(mouse_pos):
           self.selected_inventory_item = item
           return

       drop_rect = pygame.Rect(item_x, DROPBUTTON_POS_Y, 50, 20)
       if (
               item == self.selected_inventory_item
               and item.movable == "yes"
               and drop_rect.collidepoint(mouse_pos)
       ):
           # Drop logic
           item.picked_up = False
           self.game.item_states[item.name] = False

           # Undo stats if applicable (prevent negative values)
           if item.stat != "none":
               current_stat = self.game.stats.get(item.stat, 0)
               self.game.stats[item.stat] = max(0, current_stat -
item.effect)

           # Assign item back to its original room
           for room_name, room_items in rooms.items():
               if any(room_item["item"] == item.name for room_item in room_items):
                   target_room_items = self.game.rooms.get(
                       room_name.lower().replace(" ", "_"), []
                   )
                   target_room_items.append(item)
                   break

           self.game.inventory.remove(item)
           self.selected_inventory_item = None
           self.game.status_message = f"Dropped the {item.name}."
           self.game.message_timer = pygame.time.get_ticks()
           return
```