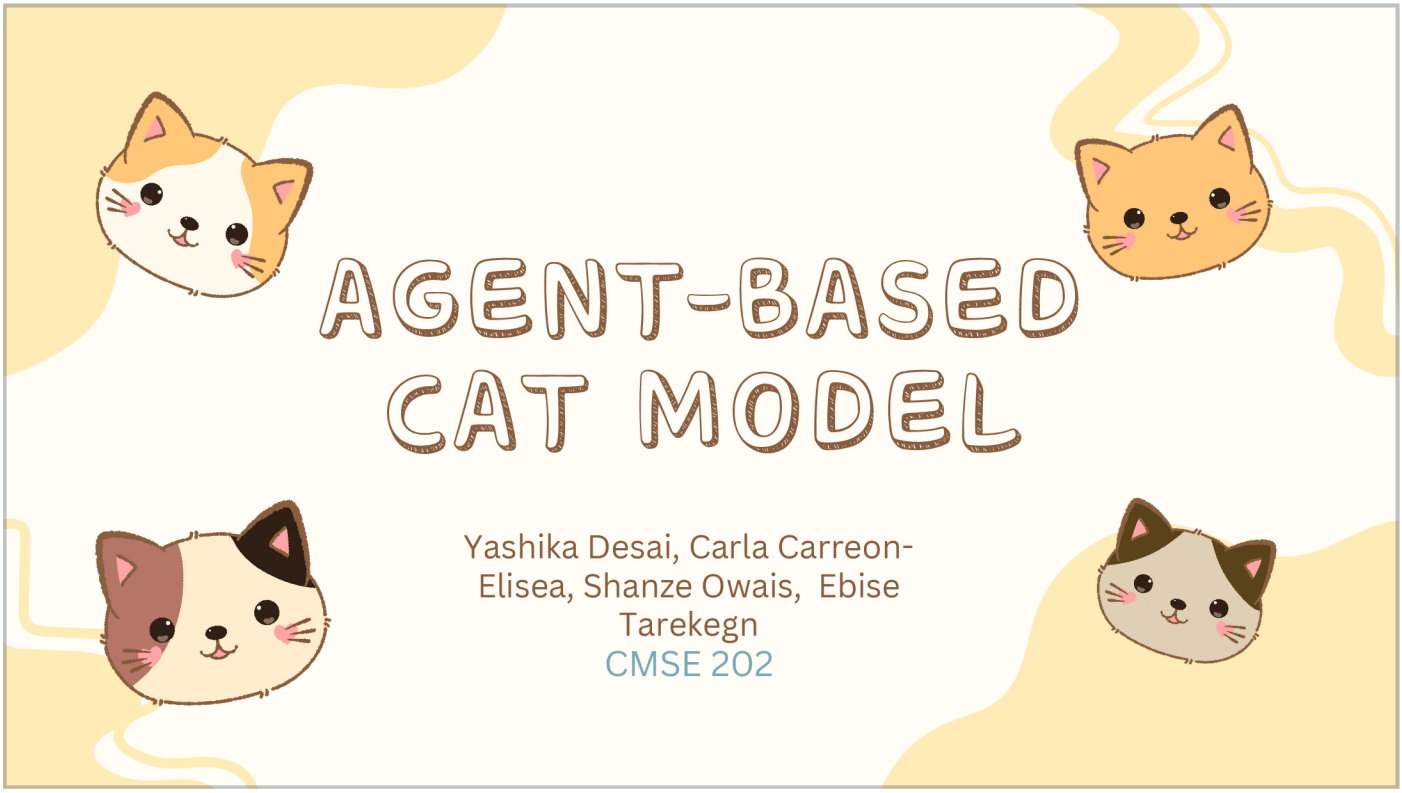

AGENT-BASED
CAT MODEL
Yashika Desai, Carla Carreon-Elisea, Shanze Owais, Ebise Tarekegn
CMSE 202

INTRODUCTION

Agent-based modeling is used to simulate complex environments to understand and mimic dynamic interactions between agents. Each environment is created with rules/parameters that dictate the behavior patterns of the agents. In this project, we sought to understand the relationship between differing cat personalities and the resulting interactions in a closed environment.

HYPOTHESES

1. Will cats of similar personalities interact with each other when their personality levels are the same?
2. Will cats of different personality traits interact negatively or avoid one another due to differences in personalities?
3. Are the cats able to maintain their needs alone without being instructed to (i.e. setting initial hunger levels)?

MEET THE CATS!

Tito

Playful

Tobi

Shy

Dumpling

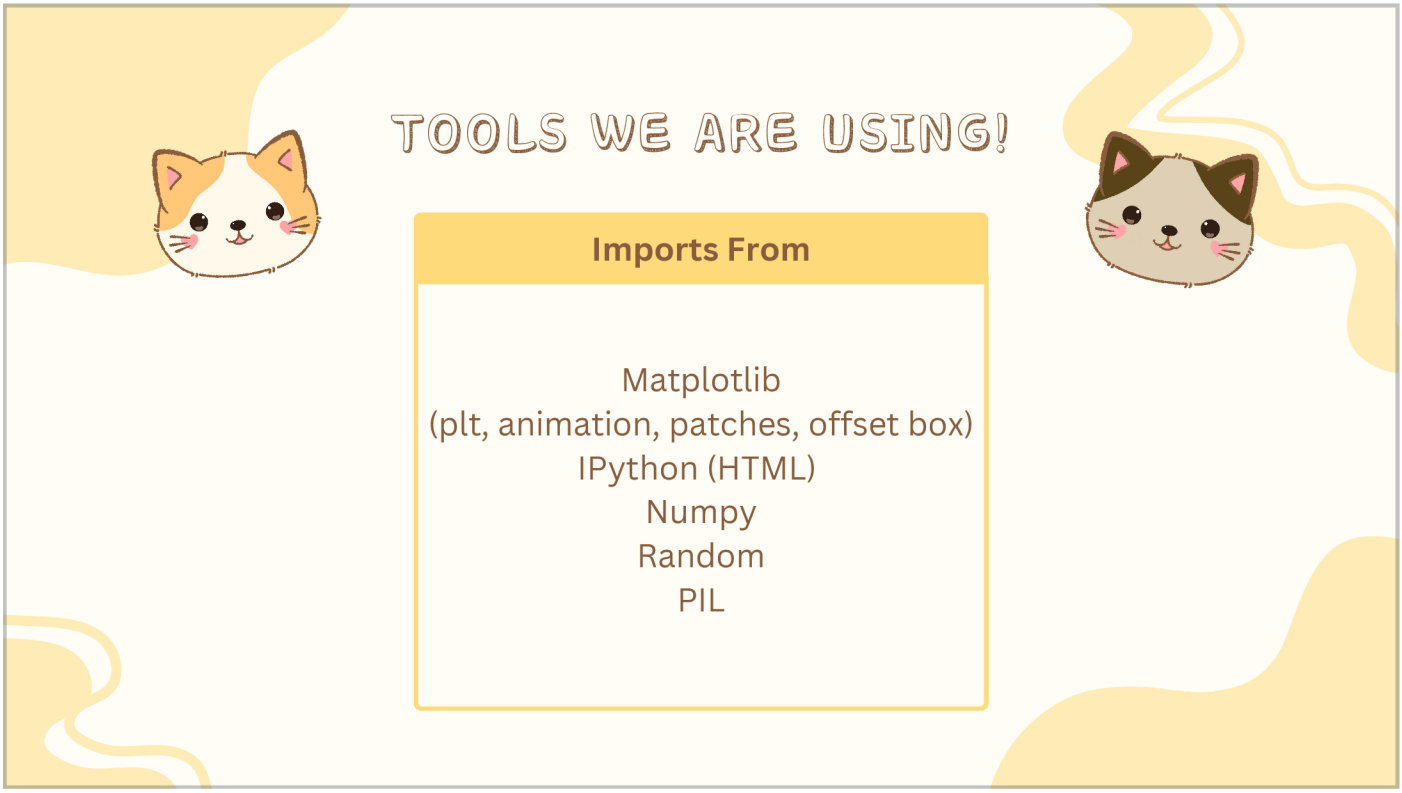
Calm

Jiji

Aggressive

We will be seeing how each of these cats interact with each other, as well as how they choose to spend their time.


VARIABLES TO KEEP TRACK OF!
Personality Traits
Calm
Shy
Playful
Aggressive
Independent Needs
Sleep
Hunger
Potty
Play
Activity
Fight
Sleep
Play
Eat
Bathroom


TOOLS WE ARE USING!
Imports From
Matplotlib
(plt, animation, patches, offset box)
IPython (HTML)
Numpy
Random
PIL

CAT DATA

We have set a legend explaining what each cat trait does/means. Giving full details

```
legend= {'Calm' : 'Very relaxed; Gets along with all cats; Gets along with all humans',
         'Shy' : 'Takes a while to warm up, hides away; Does not interact with other cats' ,
         "Playful" : 'Wants to hunt and play; Gets along with Calm cats, Fights with Aggressive cats;',
         'Aggressive' : 'Will hiss, growl, and bite; Fights with all cats;'
    }


#
# Trait Behavior Around Others
# Shy ≥ 2   Avoids other cats — moves away if another cat is nearby
# Aggressive ≥ 2 '  Approaches other cats — initiates "conflict" when nearby
# Calm ≥ 2  Doesn't react to other cats (neutral)
# Playful ≥ 2   Moves toward cats (seeks interaction), unless the other cat is aggressive
```

```
cats = {
    "Tito": {
        "position": [0, 5],
        "image": "tito.png",
        "goal": "bed",
        "personality": {'Playful': 2, 'Shy': 1, 'Calm': 1,
        'Aggressive': 1},
        "play_count": 0,
        "fight_count": 0,
        "interaction_count": 0,
        "last_interaction_hour": 0,
        "original_goal": "bed",
        "fed": False,
        "used_litter": False,
        "roam_timer": 0,
        "roam_target": None
    },
```

Data structure of the cats to track all behaviours, interactions, and the cats' state

MAP ZONE

```python
zones = {
    "bed": [(x, y) for x in range(0, 3) for y in range(5, 7)],
    "food": [(x, y) for x in range(0, 3) for y in range(0, 2)],
    "litter": [(x, y) for x in range(4, 7) for y in range(5, 7)],
    # "roam" zone is anywhere not in the other
    # and not in the litter box, bed or food zone
    "roam": [(x, y) for x in range(grid_size) for y in range(grid_size)
             if (x, y) not in [(x, y) for x in range(0, 3) for y in range(5, 7)]
             and
                (x, y) not in [(x, y) for x in range(0, 3) for y in range(0, 2)]
                and
                (x, y) not in [(x, y) for x in range(4, 7) for y in range(5, 7)]]
}
```

The Zone dict sets a structure for the environment
Setting up locations for each need

CAT ACTIONS

These functions determin certain cat interactions

- can_play determines if two cats play
- will_fight determines if two cats fight

Simulating positive and negative interactions

```
def can_play(cat1, cat2):
    if cat1["personality"].get("Aggressive", 0) >= 2 or cat2
    ["personality"].get("Aggressive", 0) >= 2:
        return False
    if cat1["personality"].get("Playful", 0) >= 2 and cat2
    ["personality"].get("Playful", 0) >= 2:
        return True
    if cat1["personality"].get("Playful", 0) >= 2 and cat2
    ["personality"].get("Calm", 0) >= 2:
        return True
    if cat2["personality"].get("Playful", 0) >= 2 and cat1
    ["personality"].get("Calm", 0) >= 2:
        return True
    return False

def will_fight(cat1, cat2):
    a1 = cat1["personality"].get("Aggressive", 0)
    a2 = cat2["personality"].get("Aggressive", 0)
    p1 = cat1["personality"].get("Playful", 0)
    p2 = cat2["personality"].get("Playful", 0)
    return (a1 >= 2 or a2 >= 2) or (p1 >= 2 and a2 >= 2) or (p2 >=
    2 and a1 >= 2)
```

Determines and updates the cat's next "goal" based on current needs & location

```
def update_cat_goal(name, info, current_hour):
    x, y = info["position"]

    # Check if cat is in target zones
    is_in_food_zone = is_in_zone(info["position"], "food")
    is_in_litter_zone = is_in_zone(info["position"], "litter")
    is_in_bed_zone = is_in_zone(info["position"], "bed")

    # Process after eating logic
    if info["goal"] == "food" and is_in_food_zone and not info["fed"]:
        # Reset interaction counts after eating
        info["interaction_count"] = 0
        info["last_interaction_hour"] = current_hour
        info["fed"] = True
        info["used_litter"] = False

        # Go to litter box after eating
        info["goal"] = "litter"
        show_goal_change_text(name, info["position"], "litter", "after eating")
        return True

    # Process after using litter box
    elif info["goal"] == "litter" and is_in_litter_zone and info["fed"] and not
    info["used_litter"]:
        info["used_litter"] = True
```

VISUAL & STATUS

Responsible for managing the simulation behaviour and FPS, making sure the process updates as needed

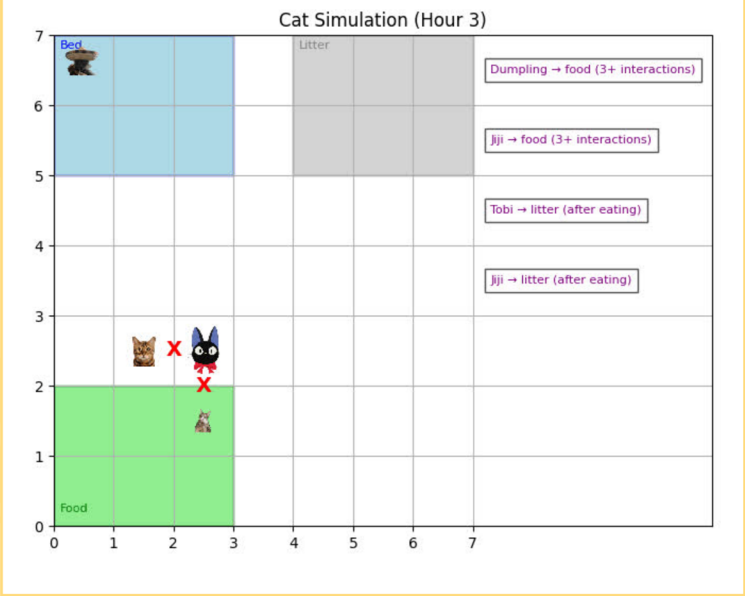
```python
def update(frame):
    global frame_counter
    frame_counter += 1

    current_hour = frame // steps_per_hour + 1
    if current_hour <= total_hours:
        title_text.set_text(f"Cat Simulation (Hour {current_hour})")

    # Remove expired goal change texts
    for i in range(len(goal_change_texts) - 1, -1, -1):
        text, expiry_frame = goal_change_texts[i]
        if frame_counter >= expiry_frame:
            text.remove()
            goal_change_texts.pop(i)
```

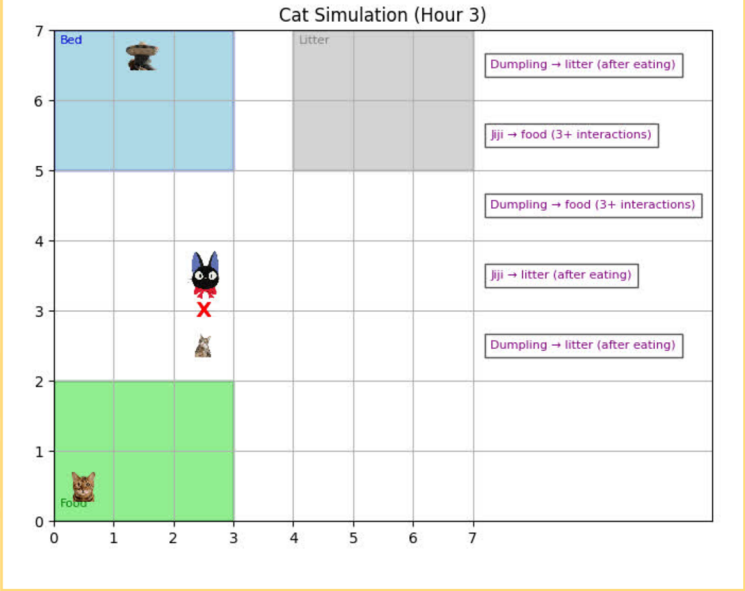
```python
def print_final_stats():
    print("Final Interaction Counts:")
    for name, info in cats.items():
        print(f"{name}: Plays = {info['play_count']}, Fights = {info
        ['fight_count']}, Total = {info['play_count'] + info['fight_count']}")
```

Provides a summary of the simulation, displaying statistics for each cat



CAT-TO-CAT VISUALIZATION#1: NORMAL PERSONALITY

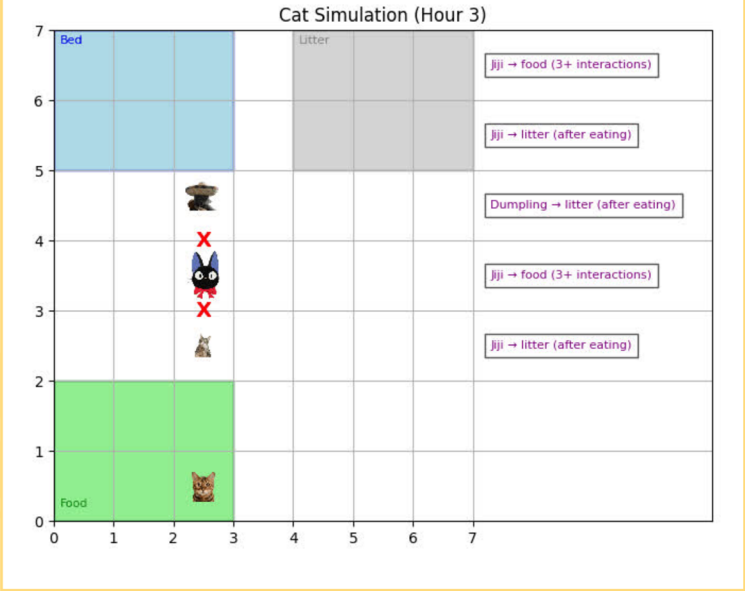
	Tito (Playful)	Tobi (Shy)	Dumpling (calm)	Jiji (aggressive)
Play	27	0	27	0
Fights	111	10	50	171
Total	127	10	77	171



CAT-TO-CAT VISUALIZATION #2: PAIRED PERSONALITY"

- TITO & TOBI: PLAYFUL = 2 & SHY = 2
- DUMPLING & JIJI: CALM = 2 & AGGRESSIVE = 2

	Tito	Tobi	Dumpling	Jiji
Play	7	7	0	0
Fights	73	29	187	121
Total	80	36	187	121



CAT-TO-CAT VISUALIZATION #3: INTENSE PERSONALITY

- SHY, CALM, PLAYFUL, & AGGRESSIVE = 2
- SHY + CALM & PLAYFUL + AGGRESSIVE

	Tito	Tobi	Dumpling	Jiji
Play	0	0	0	0
Fights	90	42	108	128
Total	90	42	108	128

CONCLUSIONS

1. PAIRED PERSONALITIES RESULTED IN PAIRED INTERACTIONS
2. INTENSE PERSONALITIES YIELDED ONLY AGGRESSIVE INTERACTIONS
3. CATS ARE SELF SUSTAINING WITHOUT COUNT VARIABLES/INITIALIZATION OF NEEDS


THANK YOU!
Any Questions?