Arriyan Ali | IAT265

Kitty the Hat Makin' Cat: Game Design Document

1. Project Description

An outdoor cat named Kitty picks her favorite hat and leaves the house for the day, she meets her best friend Darcy on her walk and engages in dialogue. Darcy tells her that she's planning a new fashion project and needs help in collecting materials to make a beautiful new hat. Kitty agrees to lend a hand and walks through the forest collecting the most perfect items for what Darcy has envisioned. When she finally meets Darcy at the end of the day, they are able to combine the objects they've collected to create a gorgeous abomination of an accessory (fit for the most regal of kittens).

2. Main Elements + Proposed Structure

Kitty (public class)

Rationale + Description: This class controls both the appearance and behaviors of the player/avatar in this simulation.

Fields and Methods:
- PVector pos; //holds kitty's position on screen
- double scale; //scalable size of kitty
- BufferedImage kitty; //store the drawn image of kitty

+ Kitty(PVector pos, double sc); //initialize pos & scale when kitty object is created
+ drawKitty(Graphics2D g2); //use kitty image (declared in constructor) to draw it within here

Environment (public class)
**renamed this class to "Layer" within first code submission

Rationale + Description: Used to create a dynamic background which moves when the right/left key is pressed. The avatar/player remains still, although this moving background creates the illusion that the avatar is moving.

Fields and Methods:
- PVector pos, vel; //holds the current position & velocity of the object
- float dampen; //this variable helps to slow down and stop movement within move() when right() or left() is not called
- double scale; //scales the image used as the environment (if needed)
- float speed; //speed at which environment moves when a key is pressed

+ Layer(PVector pos, PVector vel, double sc, BufferedImage img, float spd);

//holds the position, velocity, scale, image used as the background, and speed at which the background moves
+ right();   //environment moves left when right key is pressed (giving illusion that avatar is moving right)
+ left();   //environment moves right when left key is pressed (giving illusion that avatar is moving left)
+ move();   //environment moves based on its current velocity
+ drawEnvironment();   //display the environment

Button (public abstract class)

Rationale + Description: A superclass that any object with button-like behaviors can extend from.

Fields and Methods:
- protected double xPos, yPos, scale;
- protected BufferedImage img;

+ abstract drawButton(Graphics2D g2);   //any child of this class will need a draw method
+ boolean clicked(double x, double y);   //boolean detects whether clicked

Dialogue (public class)

Rationale + Description: During the dialogue portion, there will be multiple chat bubbles between Kitty and Darcy, this class will help display and control them efficiently.

Fields and Methods:
- PVector pos;
- double scale;
- boolean showBubble;

+ Dialogue(PVector pos, double sc);
+ drawDialogue(Graphics2D g2);   //based on whether showBubble is true, draw the dialogue bubble

CollectableItem (public class extends Button)

Rationale + Description: This class exists to define the behavior and on-screen appearance of items that Kitty collects within the forest.

Fields and Methods:
+ boolean itemStored;   //used to dictate appearance of item on screen

+ CollectableItem(double xPos, double yPos, double sc);   //declare scale, x position, and y position as parameters (variables which have been inherited by the button superclass)
- storeItem();   //this method changes the position of the button and set the itemStored boolean to true
+ drawItem(Graphics2D);   //draw the item on screen depending on whether itemStored boolean is true

DarcysHat (public class)

Rationale + Description: At the end of the simulation, the user is able to choose between items that have been collected to create a unique hat based on those selected items. This class will identify the chosen items and return a result based on them.

Fields and Methods:
- int mouse, leaf, flower, string; //store the items collected as integers, 6 possible options for a hat
- String itemOne, itemTwo;
- BufferedImage ml, mf, ms, lf, ls, fs; //different images for the final hat reveal, depending on the items chosen

+ CollectedItem(String itemOne, String itemTwo); //the two items selected as parameter
+ drawHatReveal(Graphics2D g2); //draw the image displaying the hat reveal

SpriteSheetSlicer (public class)

Rationale + Description: Since I will be using a few sprite sheets, this class will help me slice them efficiently. This class will likely exist in the utility package of the project.

Fields and Methods:
- BufferedImage image; //holds the unsliced sprite sheet

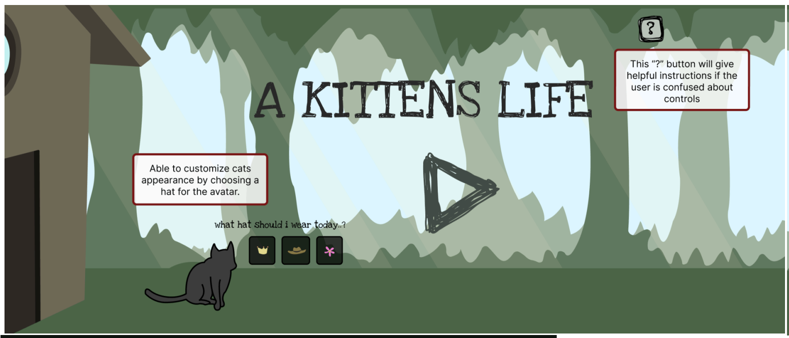
+ SpriteSheetSlicer(BufferedImage image); //have any unsliced sheet as parameter
+ BufferedImage singularSprite(int col, int row); //subdivide image depending on the row & column of the sprite I need, then return that part of the image as the BufferedImage from this method

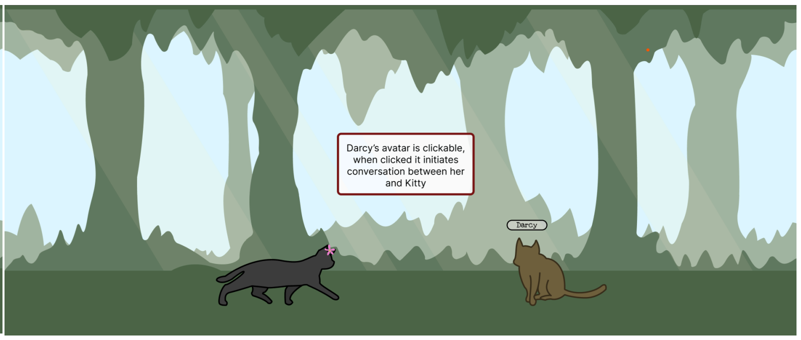
3. Execution Flow + Process

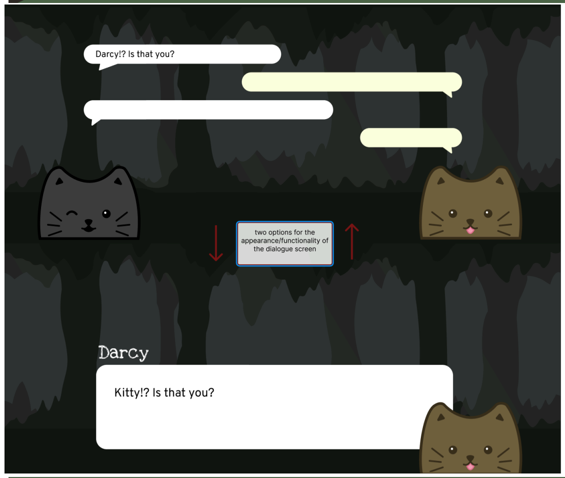
```
- int panWidth, panHeight;
- Timer t;

- Environment environment;
- Kitty kitty;
- DarcysHat finalHat;
- boolean left, right; //controls which direction Kitty "moves"
- double mouseX, mouseY; //if the user clicks the screen, these variables
hold the location of the mouse during it
- int state = 0; //controls state or appearance of the process
- StartButton startButton; //an object of a class that extends from Button
superclass, used to start the process
- RestartButton restartButton; //an object of a class that extends from
Button superclass, used to restart the process
- ConfirmButton confirmButton; //an object of a class that extends from
```

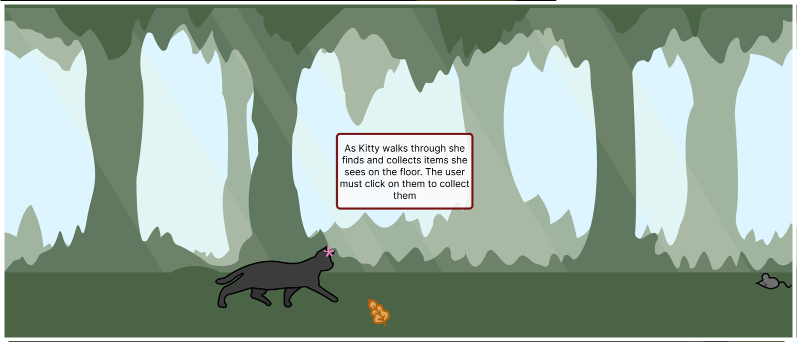
```
Button superclass, used to confirm the item choices during the selection
screen for creating Darcy’s hat

- ArrayList<CollectableItem> collectableItems; //collectable items that
have not been clicked on exist in this arraylist
- ArrayList<CollectableItem> storedItems; //when an item has been clicked
on, it is added to the stored item arraylist, since items appear
differently and trigger certain methods based on whether they have been
clicked on or not
- ArrayList<Dialogue> dialogueBubbles; //all dialogue frames or bubbles are
stored in this arraylist, dialogue may progress based on mouse clicks from
the user

+ PojectPanel(JFrame frame); //initialize all variables and their
parameters

+ paintComponent(Graphics g); //paint buttons, avatars, items, and screens
depending on the current state

+ actionPerformed(ActionEvent e) //objects and events are animated here
//Different events for each state:
//state 0: starting screen
//state 1: normal background and avatar movement/interactions
//state 2: dialogue screen
//state 3: item selection screen
//state 4: hat reveal screen

+ class MyMouseListener() //an inner class that helps detect mouse input,
specifically when the mouse clicks on a button or object

+ class MyKeyAdapter() //an inner class that helps detect keyboard input, I
will be using the A and D keys to move left and right
```

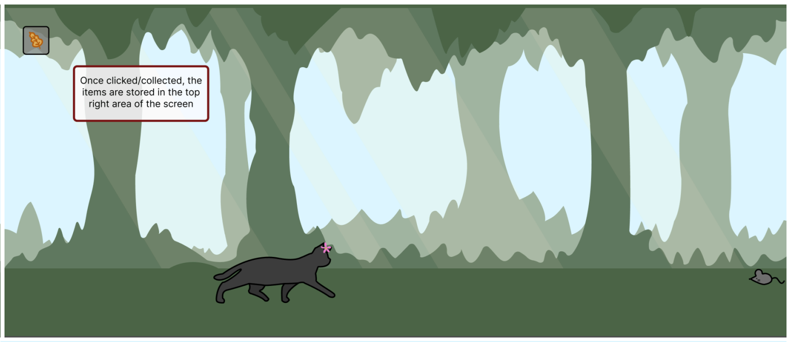
4. Illustrations + StoryBoard

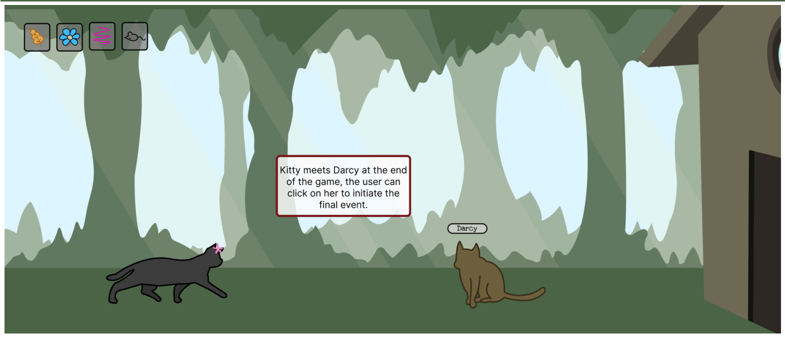

A KITTENS LIFE
?
This "?" button will give helpful instructions if the user is confused about controls
Able to customize cats appearance by choosing a hat for the avatar.
what hat should I wear today..?

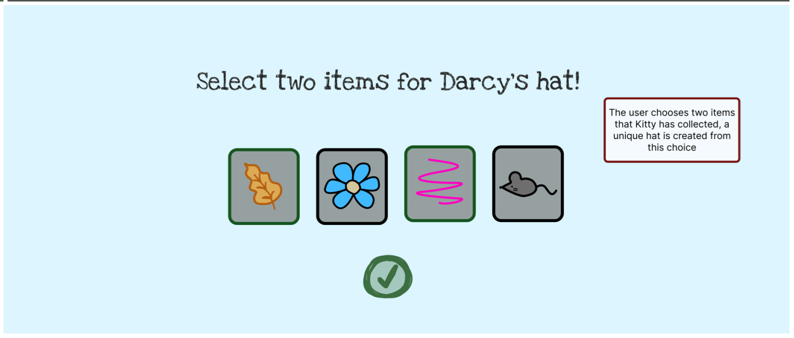

Darcy's avatar is clickable, when clicked it initiates conversation between her and Kitty
Darcy

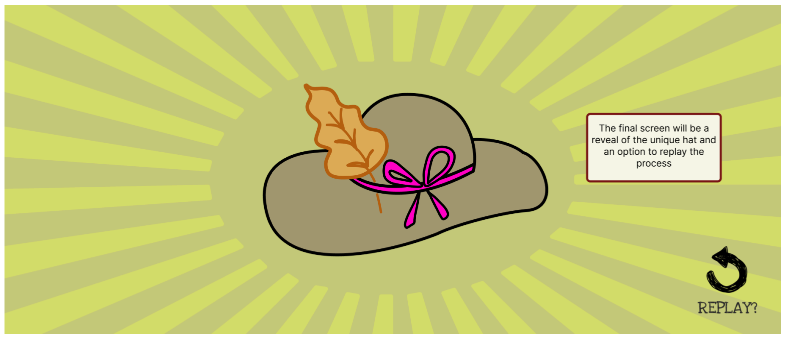

Darcy!? Is that you?
two options for the appearance/functionality of the dialogue screen
Darcy
Kitty!? Is that you?


As Kitty walks through she finds and collects items she sees on the floor. The user must click on them to collect them


Once clicked/collected, the items are stored in the top right area of the screen


Kitty meets Darcy at the end of the game, the user can click on her to initiate the final event.
Darcy


Select two items for Darcy's hat!
The user chooses two items that Kitty has collected, a unique hat is created from this choice


The final screen will be a reveal of the unique hat and an option to replay the process
REPLAY?

5. **First Code: Objects and Implementations**

First Code Interactions & Objectives:

- Keyboard interaction to control Kitty/avatar (as well as a walking animation for Kitty)
- A moving Environment/background (may require an ArrayList if I end up overlapping multiple background images**)
- A title and clickable play button, which both disappear after the play button has been clicked

```
- int panWidth = 1488;
- int panHeight = 628;
- Timer t;
- Environment environment;
// OR - ArrayList<Environment> environments;
- boolean left, right;
- Kitty kitty;
- StartButton startButton;
- double mouseX, mouseY; //used in mouseListener to detect mouse has
clicked an area where a button exists
- boolean catWalk = false; //a boolean which controls whether Kitty’s
walking animation is playing or not
- int state = 0; //used to control which state/appearance of the process is
being shown


+ PojectPanel(JFrame frame); //initialize all variables and their
parameters

+ paintComponent(Graphics g); // Draw the environment, kitty. If the state
is 0, draw the startButton. Once the state is 1, remove the start button
from the screen.

+ actionPerformed(ActionEvent e) // Make sure Kitty and the background only
move if the state is equal to 1. If the “D” key is clicked, move the
environment according to its right() method. If the “A” key is clicked,
move the environment according to its left() method;

+ class MyMouseListener() //an inner class that helps detect mouse input,
specifically when the mouse clicks on a button or object
```

```
//mouseX and mouseY values are assigned here, they are reassigned everytime the mouse clicks somewhere on the window
//Specifically check if the start button has been pressed during state 0, if this is true change the state to 1

+ class MyKeyAdapter() //an inner class that helps detect keyboard input, I will be using the A and D keys to move left and right
//If “A” or “D” key has been pressed then left or right boolean is true, otherwise they are both false. If either key is pressed then the catWalk boolean is also true, which triggers the cat walking animation
```