Tamuda: tchimada@u.rochester.edu
Amanda: awong33@u.rochester.edu
Natnael:nayalew@u.rochester.edu

Tuesday/ Thursday
Lab 9:40am-10:55am

Instructions:

- Use Up and Down arrows to move player.
- Dodge the caterpillars for at least 30 seconds to win.
- You lose a life the more you collide with caterpillars

This game is a one-player game where your goal is to play until the timer goes off whilst avoiding the scary caterpillars. In this game, you are playing as a player named Rocky, the YellowJacket, the University of Rochester's mascot. When the game is first played, you will start in the middle left of the screen and start playing by interacting with the keyboard, which is in this instance by pressing the **up or down arrow** keys. Once the game has started, your goal is to avoid the obstacle, the caterpillars. If not, you will lose a life. Each player starts with three lives and if you stay in contact with a caterpillar for more than one second, you lose a life. Therefore, when you hit zero lives, you will lose the game. However, you win the game when the timer hits zero and you have at least one life remaining. The "you win" label will be displayed when you win accompanied with a victory sound.

The requirements of the game were for it to be animated, interactive, display a scoring mechanism that is shown to the player, have a definitive ending mechanism, have collision detection, and have a flourish per team member.

1. If **animation** is defined as a visual or image that shows movement, our game includes animation. While our game is in progress, the bee's wings flap up and down. Essentially, when the game is on, the bee is animated because the bee is in motion.

2. If **interactive** means that the game has to involve using the keyboard or mouse, our game is interactive. In our game, to move the bee to be in the position you desire, you have to use the keyboard up arrow and down arrow keys to move.

3. If the **scoring mechanism** that is shown to the player means that the score of the player has to be seen during the game, then our game includes it. The scoring mechanism is shown on the top right of the screen through the number of lives the player has also seen as the bees.

4. If a **definitive ending mechanism** means that the player can win or lose, then our game incorporates it because you will lose you have no remaining lives. You will win; however, if you are able to play for at least 30 seconds. Additionally, when you lose after the timer stops, you will hear a losing sound.

5. If a **physical mechanism** is defined as incorporating velocity, mass, motion, or gravity, then our game incorporates a physical mechanism due to the continuous motion required to keep the bee in a horizontal straight line. Essentially, after about one second of nonexistent interaction with the keyboard, the bee will start to go down, therefore incorporating gravity.

6. If **collision detection** is defined as incorporating a hazard, reward, or item, then we incorporated collision detection due to the loss of lives if you were to collide with a caterpillar. Thus, making the caterpillar a hazard. Additionally, when the bee collides with a caterpillar, the bee transforms from a stereotypical yellow bee to a unique orange unhealthy bee.

7. If flourishes mean components of our project that make it unique, then we have included three flourishes for each of our teammates. If including **randomized stationary obstacles** means the object has to be destroyed by an object randomly, then we have incorporated it due to the randomized position of caterpillars at the start of each game. Thus, the unique positions of the caterpillars at the start of each game make it randomized. If sound effects mean that the game plays **sound** while playing, then we have incorporated sound because when the game is on, it plays an uplifting, childish song. If making it look cool means that you have to have learned about polygon filling and textures to make the game actually look good, then we have incorporated it because we have gray and blue clouds in our background.

Outline:

a) In our **BeeGame class**, this is where we set the size of our window and JFrame. In this class we specified that we wanted the size of the window to be 800 by 800.

b) In the **Renderer class**, this is where we draw all the components on the screen. We use the update and repaint to redraw the components very 10 milliseconds with the help of a game thread.

c) In the **KeyHandler class**, this is where we specified what keys players would interact with. In our game, we used KeyPressed and KeyReleased to have the bee go up or down when that arrow key was pressed.

d) In our **sound class**, we construct the sounds played in the game. As previously mentioned, there is a an uplifting, childish song while playing. However, depending on the outcome of the game (the definitive mechanism), there are different soun effects that are played and they are all initialized in the sound class.

e) In the **UI class**, this is where we incorporated the timer that appears on the top right of the screen. Additionally, we have added another component that displays the end results of the game, the “you win” and “you lose” aspect. The getLifeImage() makes displays the number of lives on the top right as well.

f) The **player class** defines the bee, main character of the game. Here we initialize the different sprites that represent the state of the bee (flying, colliding, gliding), we also construct an invisible rectangle (r1) that is used for collision detection with (r2) another invisible rectangle that is built around all caterpillars. The update method here is responsible for changing the bee’s position while an arrow key is pressed. The bey’s location y is incremented by the “speed” each time an arrow key is pressed.

g) The **Worm class** creates and randomizes the caterpillar. Both the number of caterpillars appearing on screen and their speed are randomized. Once the caterpillars reach are out of view on the left side, they are disposed and a new group of caterpillars is created on the right. Additionally, this is where the updates on the caterpillar’s position gets updated such as when it goes up and down.