**SHUTTER BUG GAME**

**Made by:**

**1. Ahmed Mohamed Fawzy 221903175**

**2. Passant El-Tonsy Ali 221903220**

**3. Nardeen Zakaria Francis 221903121**

**4. Mohamed Yasser Ismaeel         221903109**

**5. Merna Badr Ibrahim 231903556**

**Supervised by:**

**Dr. Shimaa Ibrahim**

**Eng. Mohamed Abdellatif**

**Contents**

1. **Overview……………………………………………………….…3**
2. **Phases of Project…………………………………………………3**
3. **Game Logic……………………………………………………….3**
4. **Game Features and Tools………………………………………..8**
5. **Challenges and Solutions……………………………………….10**
6. **Future Enhancement…………………………………………...10**
7. **Resources………………………………………………………..11**
8. **Overview**

The Shutter Bug project is an interactive 2D game developed in Processing. The main objective of the game is to capture bugs by clicking on them within a limited timeframe while avoiding specific sick bugs that cause failure of game. The game features attractive visuals, dynamic gameplay, and a great user experience through animations and background music.

From shutter bug game objectives is that it can create a fun and interactive game to practice game development concepts, incorporate animations, transitions, and visual effects, demonstrate the use of object-oriented programming concepts, such as classes and inheritance, and enhance user experience with sound effects and visual elements.

1. **Phases of Project**

**2.1. Design and Planning:** this phase focuses on conceptualizing the game by sketching, defining the arrangement of bugs, viewfinder, score display, and timer, choosing a theme, defining main game logic, and movements.

**2.2. Collecting Data:** this involves gathering all necessary assets for the game as images, backgrounds, fonts, audios.

**2.3. Implementation of Level One:** develop the first playable level of the game.

**2.4. Game Enhancement (Level Two):** add a second level, incorporating features to increase difficulty and distinguish it from the first level.

**2.5. Developing Level Transition and Completion Screens:** design the transition screens between levels and the final completion screen.

**2.6. Welcome Screen Implementation:** create an engaging opening screen for the game.

**2.7. Integration:** combine all game components into a cohesive system.

**2.8. Debugging:** identify and fix errors or inconsistencies in the game.

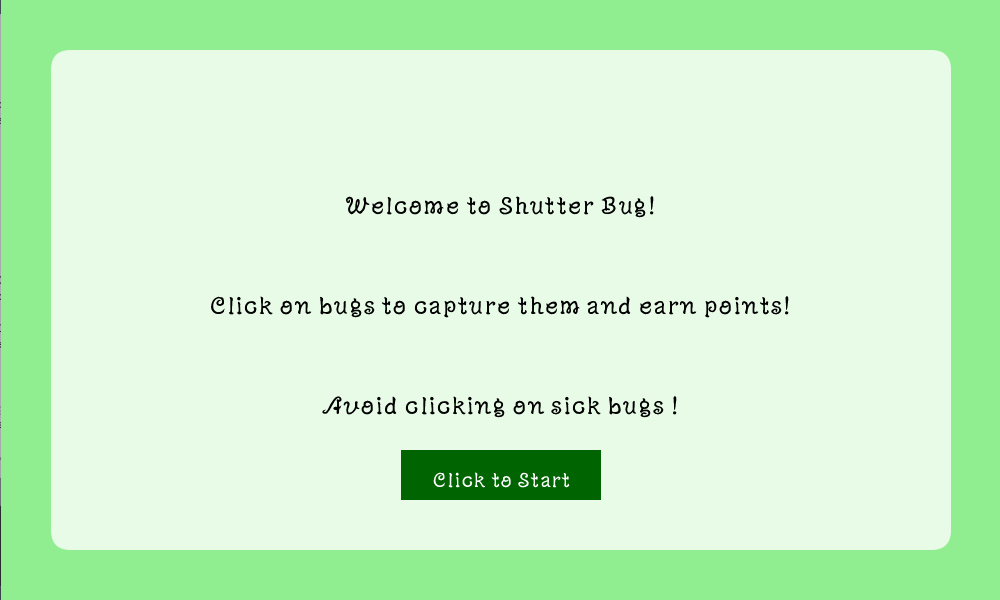
**2.9. Refining and Testing:** polish the game and test it thoroughly.

1. **Game Logic**

The idea of shutter bug game is easy to deal with it. First,a starting screen displayed with a title "SHUTTER BUG" and moving text ("Press any key to start").



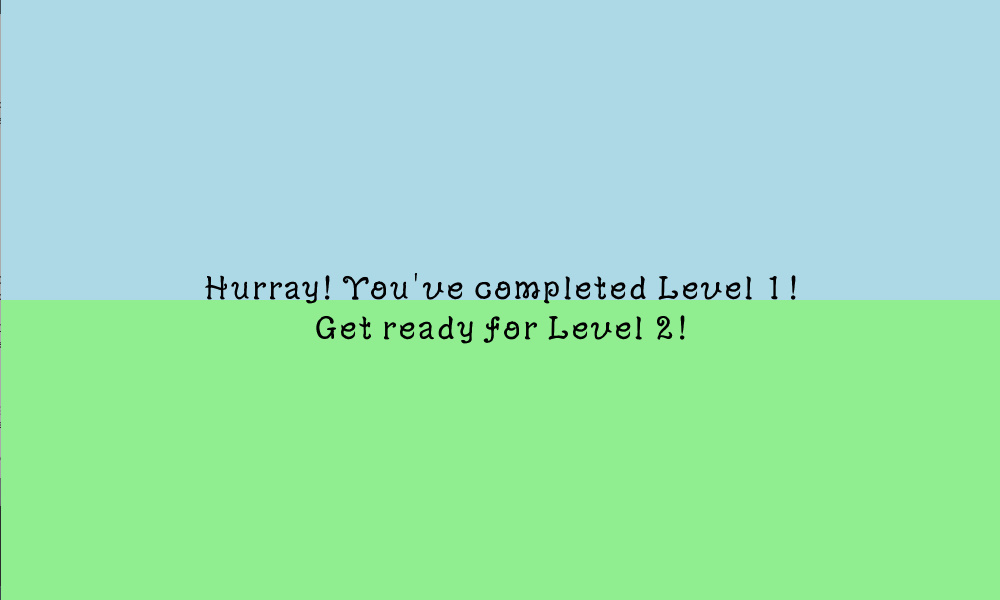
Guide Screen that provides instructions for gameplay. The player starts the game by clicking a button.



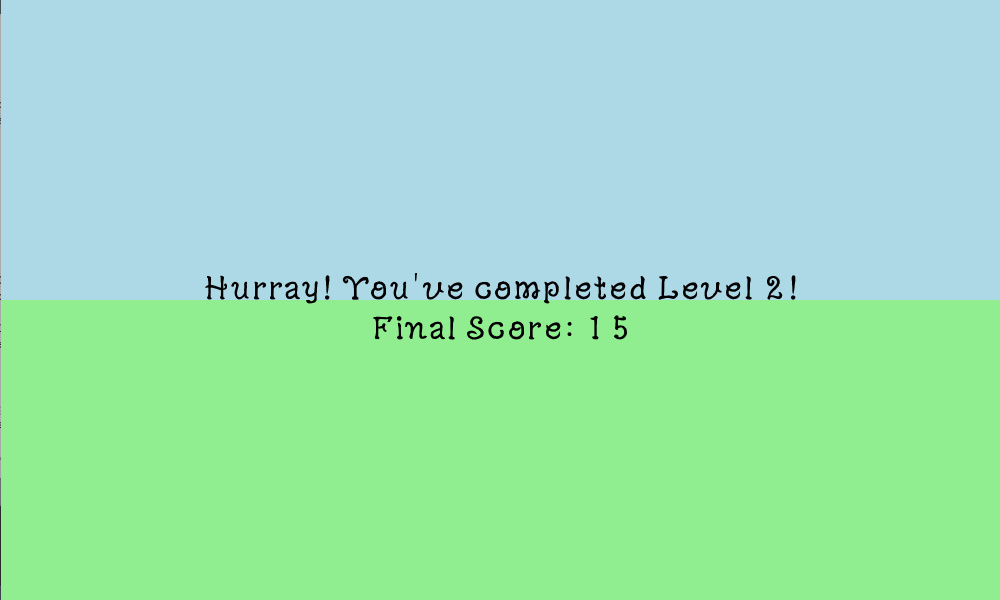
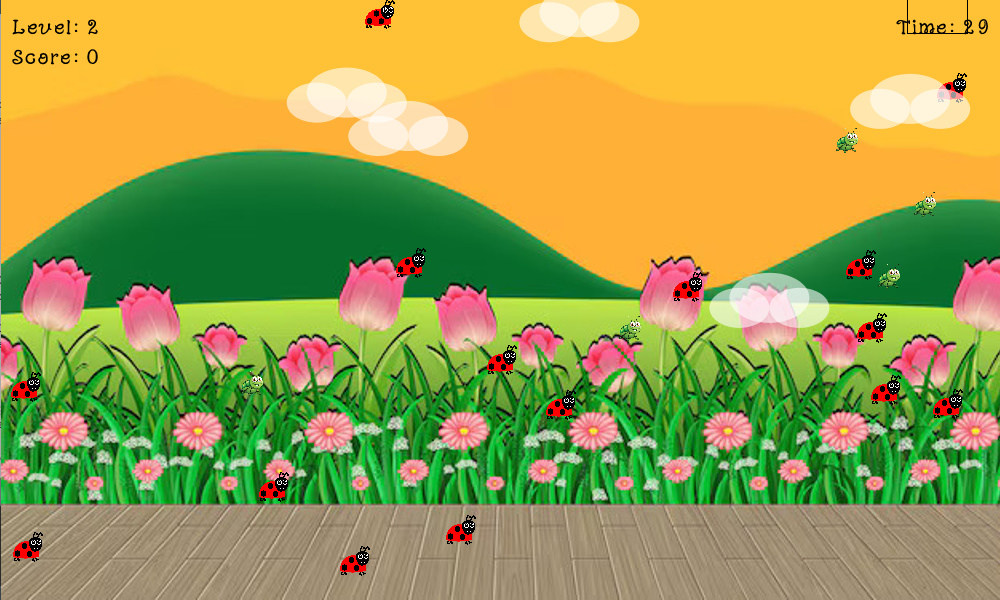
Main Game Loop controlled by the draw() function. Depending on the state (showWelcomeScreen, showGuideScreen, gameStarted.), different screens are displayed.

Then introduced Level 1 at which the player captures regular bugs by capture 10 bugs within 30 seconds to progress, and each bug moves randomly across the screen. Transition to Level 2 happens when the score reaches 10 within time limit.





At Level 2 Introducing "sick bugs." Capture 15 bugs while avoiding sick bugs. Clicking on a sick bug ends the game. Capturing these ends the game.





Each level has a countdown timer. The game ends if the timer reaches zero. Capturing bugs increases the score but capturing sick bugs (in Level 2) ends the game.

Game Over is triggered when the timer runs out or the player captures a sick bug. While completion is when the game displays a completion message after successfully finishing Level 2 with rotation of final score.



1. **Game Features and Tools**
   1. **Gameplay Mechanics:** Timer counts down from 30 seconds for each level, and ends the game if time runs out. Score increases when a bug is successfully captured and resets in Level 2 for added challenge. Viewfinder which is dynamic rectangular area centered around the mouse pointer.
   2. **Tools Used:** Processing for creating visuals and interactions. Minim Library for background music integration.
   3. **Classes:**

* **Bug Class:** Represents regular bugs with random movement.
* **SickBug Class:** Inherits from the Bug class and represents dangerous bugs in Level 2.
* **Cloud Class:** Simulates drifting clouds for visual appeal.
  1. **Core Functions:**
* **setup():** Initializes the game, loads assets, and sets up objects.
* **draw():** Contains the game loop to render the game states (welcome, guide, game).
* **gameStarted():** Handles gameplay, including bug movement, collision detection, and score updates.
* **captureBugs():** Checks if a bug is in the viewfinder and updates the score or ends the game if necessary.
* **showWelcomeScreen():** Initial screen with a moving bug and a title then press any key to the guide screen.
* **showGuideScreen():** Show guidelines and click the mouse to start the game.
* **startLevelTransition():** Move from level 1 to level 2 when you pass level 1.
* **showLevelTransition():** Deals with flow of levels.
* **showCompletionMessage():** End of game with Hurray message.
* **gameOver():** End of the game because player losses
* **size(width, height):** Sets the size of the game window.
* **millis():** Time Tracking.
* **rect(x, y, w, h):** X and y are the top left corner to start drawing the rectangle.
* **image(img, x, y, w, h):** Displays an image.
* **text(text, x, y):** Text at a specific position.
* **random(low, high):** generates random numbers or coordinates.
* **noLoop():** Stop repeating the function.
* **mousePressed:** Detect the mouse to be pressed.
* **keyPressed:** Detect the key to be pressed.
* **noStroke():** Disables outlines for the shape.
  1. **Variables:**
* **mouseX and mouseY:** are built-in variables in Processing that represent the current horizontal and vertical position of the mouse pointer.
* **speedX and speedY:** are variables that represent the rate of change in the x and y positions of an object.

1. **Challenges and Solutions**

We’ve faced some issues during the implementation and we’ve worked on solving them such as:

1. Integration phase.

2. Timer has been started in the welcome screen not the main pages.

3. Images needed to be converted to jpg and png.

4. There was a glitch while translating between welcome and guide screen.

5. There were some errors with dimensions.

6. Implementing view finder using translate function redraw the rectangle everywhere.

7. Smooth animations for bugs and transitions.

8. Managing objects (bugs and clouds) efficiently.

9. Ensuring engaging gameplay across levels.

We try to develop some solutions to decreases this challenges as we use mathematical functions like sin() for natural movement and controlled transitions with timers, use ArrayList to dynamically add and remove objects, and add difficulty through "sick bugs" and a reset score in Level 2.

1. **Future Enhancement**

Our game will not stop at this progress but we will try to develop it more and increase its features to be more interested like: sound effects for losing and winning can be added, mute option can be added, bugs can be hidden for amount of time then appear again, collecting coins through capturing bugs, adding more levels with increasing difficulty, introducing power-ups and obstacles, implementing a high-score leaderboard, adding a pause and resume functionality.

1. **Resources**

* Vividfax for searching for ideas: <https://vividfax.itch.io/>
* Google Fonts: <https://fonts.google.com/>
* Sound Effects: <https://pixabay.com/sound-effects/search/fishing/>
* Processing Documentation: <https://processing.org/reference/size_.html>
* Lab Tutorials.
* AI Tools for debugging and fixing some errors.
* Images and backgrounds (Uploaded in data folder)