
SLUG
CHASE!

# STICKY SLUG TRAIL: STICKER FEAST!

In this game, inspired by the classic "Snake," players guide a sticky, squishy slug as it slides across the screen, Eating trash that extend its gooey trail. Each trash piece adds a unique look and effect to the trail, challenging players to maneuver the slug through obstacles while building the longest, most colorful goo path possible.

# game overview

**trash Collection Madness:**
trash appear randomly across the map, each one adding a different color or effect to the slug's goo. Players can collect themed trash that may offer bonus points or special effects, like speed boosts or slowdown powers.

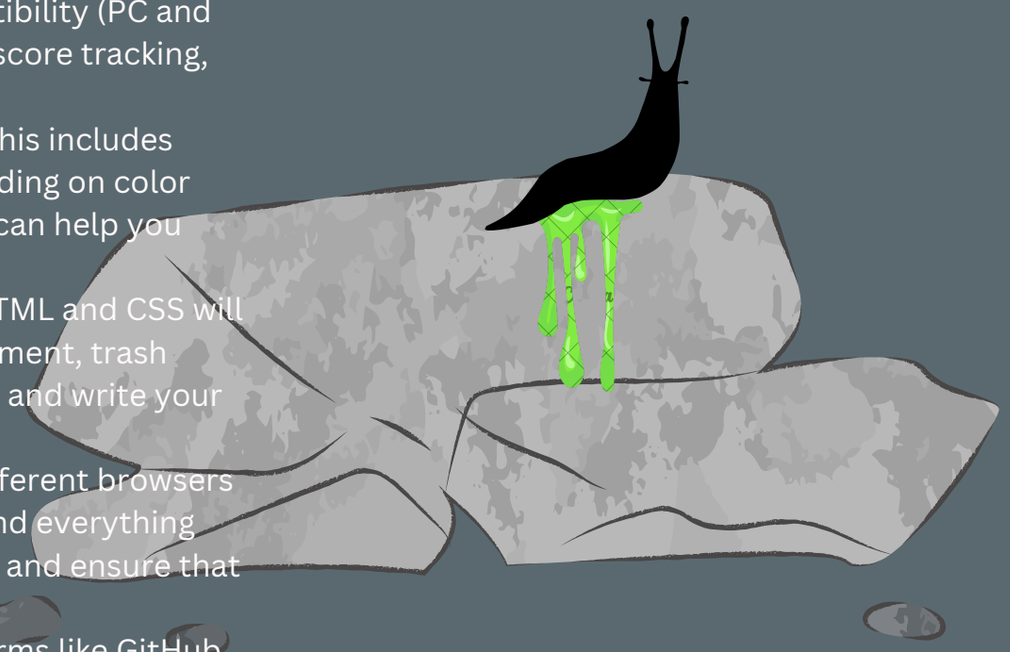
**Goo Trail Growth:**
Players control a slug that leaves behind a goo trail, which grows longer with each sticker collected. The goo trail is sticky, adding a unique challenge as players avoid their own path and navigate obstacles.

**Challenging Levels & Obstacles:**
As the slug's goo trail grows, players must skillfully maneuver around tricky layouts and avoid colliding with their own trail. Advanced levels introduce obstacles and hazards, testing players' reflexes and strategy.

10.1 Implementing the System Development Life Cycle (SDLC)

The System Development Life Cycle (SDLC) is a structured approach to software development that ensures quality and consistency throughout the project. For your slug game web page, here are the key phases:

1. Planning: In this initial phase, you'll outline the purpose of the game (slug eating trash, growing a sticky trail) and determine the project requirements, such as the game mechanics, design, and platform compatibility (PC and mobile browsers). You should define the target audience, platform (web), and the features like score tracking, movement mechanics, and collision detection.
2. Design: During the design phase, focus on creating the game's visual and functional design. This includes wireframing the layout (where elements like the game area, slug, trash, and score will go), deciding on color schemes, and how the slug and its trail will appear. Prototyping tools like Figma or Adobe XD can help you visualize how the game will look and feel.
3. Implementation: In this phase, you will start coding the game using HTML, CSS, and JavaScript. HTML and CSS will handle the layout and styling, while JavaScript will control the game logic such as slug movement, trash collection, trail growth, and collision detection. Use IDE tools like Visual Studio Code to organize and write your code, and integrate libraries like p5.js for animations if needed.
4. Testing: After building the initial version, thorough testing is critical. Playtest the game across different browsers (like Chrome, Firefox, and Edge) and devices (PC and mobile) to make sure there are no bugs and everything functions smoothly. Check for performance issues, especially with animations like the sticky trail, and ensure that the game is responsive and works well on both desktop and mobile.
5. Deployment: Once the game is complete and thoroughly tested, you will deploy it online. Platforms like GitHub Pages, Netlify, or your own server can be used to host the game. Ensure that it is accessible, that the loading times are minimal, and that users can easily start the game with a simple user interface.
6. Maintenance: After deployment, gather user feedback and fix any bugs or issues that arise. Also, consider adding additional features or improvements based on user experience (UX) insights. This could include adding new levels, trash types, or even power-ups.



10.8 Visualizing as a Problem-Solving Technique

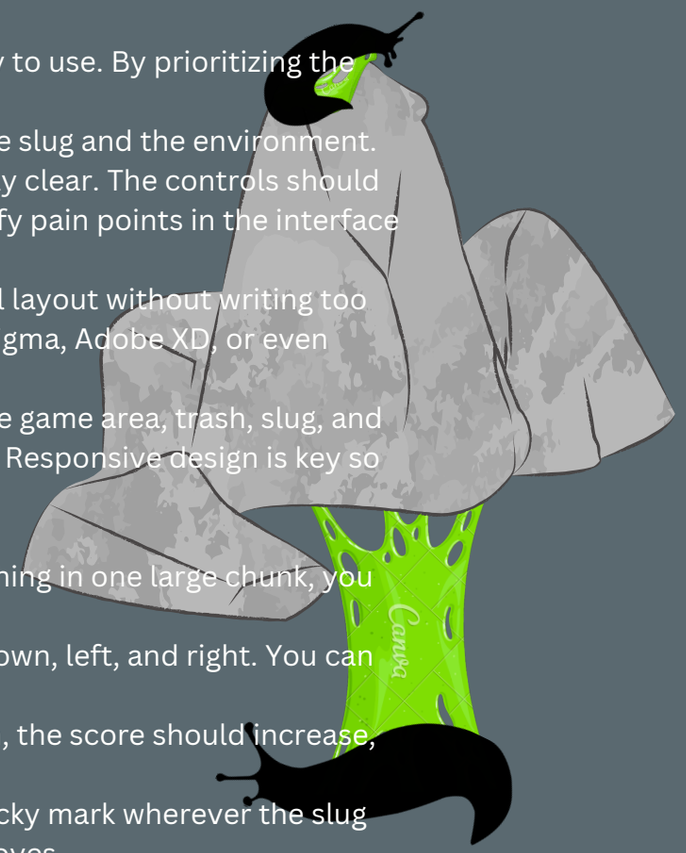
Visualization techniques like flowcharting and wireframing are essential in the early stages of the development process to help you plan out the game logic and interface before writing the code.

- Flowcharts: These are graphical representations of the game's logic and functionality. For example, you can create a flowchart that represents how the slug moves across the screen, how it collects trash, and how the sticky trail grows. This helps you understand the sequence of events and decisions that happen in the game.
- Wireframes: Wireframes will help you plan out the user interface (UI). Sketch the layout of the game's main screen, showing where the slug will move, where the trash will appear, and where the score and other information will be displayed. Tools like Figma or Adobe XD can help you create interactive prototypes to see how the game will look and function before starting the actual development.
- IDE Tools: Many IDEs, like Visual Studio Code, come with extensions that allow for live previews of your project. This enables you to see your progress and visualize how the game will appear in real-time while you're coding, providing immediate feedback and reducing potential mistakes.

15.1 Investigating User-Centered Design (UCD), Prototyping, and Wireframing

In the design process, focusing on User-Centered Design (UCD) ensures that the game's interface and interactions are intuitive and easy to use. By prioritizing the player's experience, you'll create a game that is both fun and accessible.

- UCD: User-centered design places the player at the core of the development process. Think about how the player will interact with the slug and the environment. For example, the slug should move smoothly, the trash should be easy to see and collect, and the growing sticky trail should be visually clear. The controls should also be easy to use on both desktop (keyboard) and mobile (touchscreen) platforms. Playtesting with users early on can help you identify pain points in the interface or game mechanics.
- Prototyping: Prototyping is an important part of the design phase. Creating prototypes helps you test the game mechanics and visual layout without writing too much code. This step is about testing your ideas and getting feedback before committing to development. You can use tools like Figma, Adobe XD, or even interactive HTML/CSS to create prototypes.
- Wireframing: Wireframes serve as the blueprint for the game's UI. Design the layout to be simple and intuitive. Key components like the game area, trash, slug, and score should be clearly displayed. For example, ensure the score is visible at the top, and the slug and trash are easily distinguishable. Responsive design is key so that the game works well on both desktop and mobile devices.




10.7 Decomposing a Large Programming Problem into Smaller Procedures

Breaking down the overall game into smaller, manageable tasks is a key part of effective software development. Instead of coding everything in one large chunk, you can focus on individual modules.

1. Slug Movement: Create a function that handles the slug's movement across the screen. This should include controls for moving up, down, left, and right. You can use JavaScript's keydown event listeners to capture keyboard input or touch events for mobile devices.
2. Trash Collection: Write a function to handle the appearance of trash and the collection process. Whenever the slug collides with trash, the score should increase, and the trash should disappear or reappear in a new location.
3. Trail Growth: Implement the growing sticky trail feature. When the slug eats trash, the trail should grow in length, leaving behind a sticky mark wherever the slug moves. This will require managing an array of coordinates that represent the trail and updating it as the slug moves.
4. Collision Detection: This module checks if the slug has collided with itself or the game boundaries. If a collision occurs, the game ends, and a "Game Over" message appears with the score.
5. Score and Game Over: Keep track of the score in a variable and display it on the screen. Implement a game-over condition where the player can restart after the slug collides with its own trail or the boundary.

11.13 Using Pseudocode or Graphical Representations to Plan the Program

Pseudocode or graphical representations like flowcharts help you visualize the structure of your program before you start coding. Here's an example pseudocode for your slug game:

text

Copy code

Start Game

Initialize game state (slug position, score, trail)

While game is running:

Handle user input for movement (keyboard or touch)

Move the slug according to input

If slug eats trash:

Increase score

Grow the trail

If slug hits itself or boundary:

End game and display score

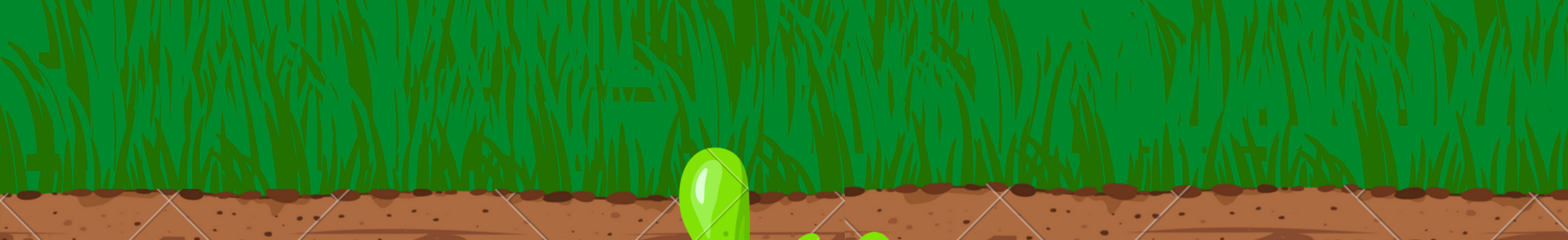
Display final score

15.7 Apply UI/UX Design for Multiple Platforms

For a seamless game experience across multiple platforms, UI/UX design should focus on accessibility and adaptability. Ensure that your game's design is responsive, which means it should adjust to different screen sizes and devices (computers, tablets, mobile phones).

- Mobile: Use touch-friendly controls like swipe gestures for movement. The interface should be clear and easy to navigate on smaller screens, with the game area and controls being large enough to interact with.
- Desktop: Ensure that the game is playable with a keyboard (arrow keys or WASD) and that the game interface is properly scaled for larger screens. The layout should adapt without losing functionality or aesthetic quality.

By following these guidelines, you'll ensure that the game is enjoyable for users on different platforms while also adhering to the principles of user-centered design.




THANKS