Project Report: CSS-Only Click Game

1. Introduction

The CSS-Only Click Game is a simple web-based game where the player has to click a moving target before time runs out. The game is designed to run entirely in the browser, using HTML for structure, CSS for layout and animation, and a small JavaScript snippet to make the game harder (increase target speed) each time you win a level.

This project demonstrates that you can build basic interactive games using mainly HTML and CSS, and enhance them with minimal JavaScript when needed.[css-tricks]

---

2. Features and Flow

- Animated Arena: A playing area shows the moving target.
- Goal: Click the target before the timer drains.
- HUD (Heads-Up Display): Shows instructions and a visual timer.
- Winning Screen: Pops up when you succeed.
- Next Level: Speeds up the target movement; makes the game harder.
- Restart Option: Lets you replay the game.

---

3. Code Explanation & Important Lines

Layout and Arena

- The entire game is inside , styled with a border and rounded corners, and a nice gradient background.
- The main area is defined by CSS variables for flexible size control:

css

```
:root {
  --arena-w: 320px;
  --arena-h: 420px;
  --target-size: 46px;
  --move-time: 4s; /* Target movement speed */
  --timer: 8s;    /* Game timer length */
}
```

Timer Bar

- Visual timer is styled as a horizontal bar:

css

```
.timer::before {
  animation: drain var(--timer) linear forwards;
}
@keyframes drain { to { transform: translateX(-100%); } }
```

The timer bar shrinks over 8 seconds, showing the time remaining to click.

Moving Target

- Target is a circular label with a gradient fill, animated with CSS @keyframes roam:

```
css
.target {
  animation: roam var(--move-time) ease-in-out infinite;
  cursor: pointer;
}
@keyframes roam {
  /* Moves target to different positions in arena */
  0% { left: 10px; top: 10px; }
  25% { left: 240px; top: 50px; }
  50% { left: 40px; top: 150px; }
  75% { left: 200px; top: 300px; }
  100% { left: 10px; top: 10px; }
}
```

The target animates along a path inside the arena.

- Clicking the target toggles a hidden checkbox (#win), which triggers the win overlay using pure CSS:

```
css
#win:checked ~ .overlay.win { opacity: 1; pointer-events: auto; }
```

Winning and Level System

- When the player clicks the target, a win screen appears via the CSS above.
- The "Next level" button is a label for the same checkbox, and also has a JS click event:

```
javascript
btn.addEventListener('click', (e) => {
  let speed = getComputedStyle(root).getPropertyValue('--move-time').trim();
```

```
  let currentSpeed = parseFloat(speed);
  if (currentSpeed < 1) { btn.hidden = true; }
  else {
    let newSpeed = currentSpeed - 1;
    root.style.setProperty('--move-time', newSpeed + 's');
  }
});
```

This JavaScript:

- Gets the current speed,
- Decreases it by 1 second (making target move faster),
- Hides the button if minimum speed is reached.

---

4. Summary of Game Logic

- Initial Play: Target moves at normal speed, timer counts down.
- Win: Click moving target before timer ends.
- Next Level: Each win, target moves faster, making the game more challenging.
- Restart: Reloads the game at starting speed.

---

5. Strengths and Limitations

Strengths:

- Game logic (click, win, timer, levels) handled almost entirely via CSS and HTML.
- Minimal JavaScript is used only for increasing difficulty.
- Fully responsive and visually engaging.
- Good demonstration of CSS animation and checkbox hack for interactivity.css-tricks

Limitations:

- Game logic is basic, score is not tracked.
- Timer bar is visual only; doesn't stop the game if time runs out (could be improved).
- Only one moving target per level.

---

6. Suggestions and Expansion

- Add more targets or obstacles for advanced levels.

- Track and display score or time left.
- End game or show "Game Over" if timer runs out.
- Add sound or effects on win.css-tricks

This is a great example project showing that simple games can be built with mostly CSS and HTML logic.