TILL DEATH DO US APART

GAME OVERVIEW

Till Death Do Us Part is a whimsical tower defense game I made for my girlfriend's 20th birthday. She loves tower defense games, so I decided to create this mini version just for her. You play as a loving boyfriend preparing a birthday cake at home for this special day. However, a horde of ravenous rabbits outside has caught the scent and is determined to crash the celebration by devouring the cake! Players must strategically place down turrets to protect the cake from waves of hungry bunnies. As the game progresses, the rabbits become craftier, requiring quick thinking and adaptive strategies to succeed.

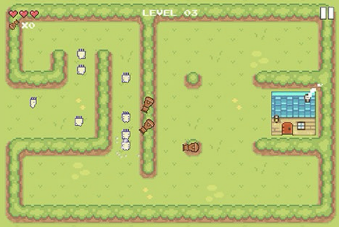
CHARACTERS

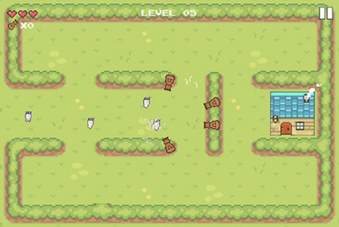
Protagonist: The loving, determined, and resourceful main character—me.

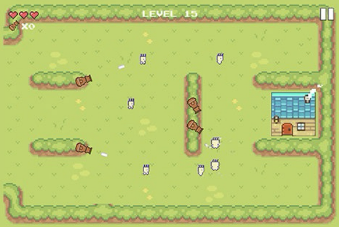
Rabbits (Enemies): rabbit is uniquely animated with distinct speed.

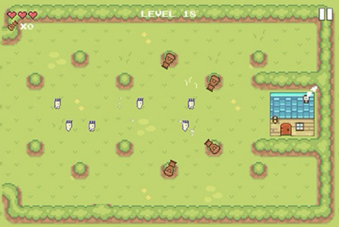
OBJECTIVES

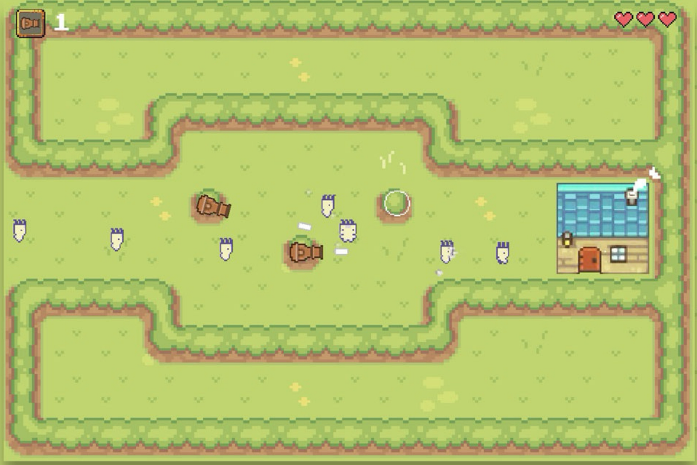
The main objective is to prevent the rabbits from reaching your house and consuming the cake. Each wave grows progressively harder, with more rabbits











GAMEPLAY CORE MECHANICS

Base Defense Mechanics: Players defend the cake by positioning a creative turrets around the garden and other entry points.

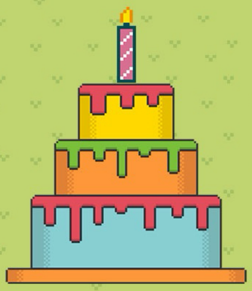
Advanced AI: Each type of rabbit has distinct movement patterns and adaptive behaviors, such as evading.

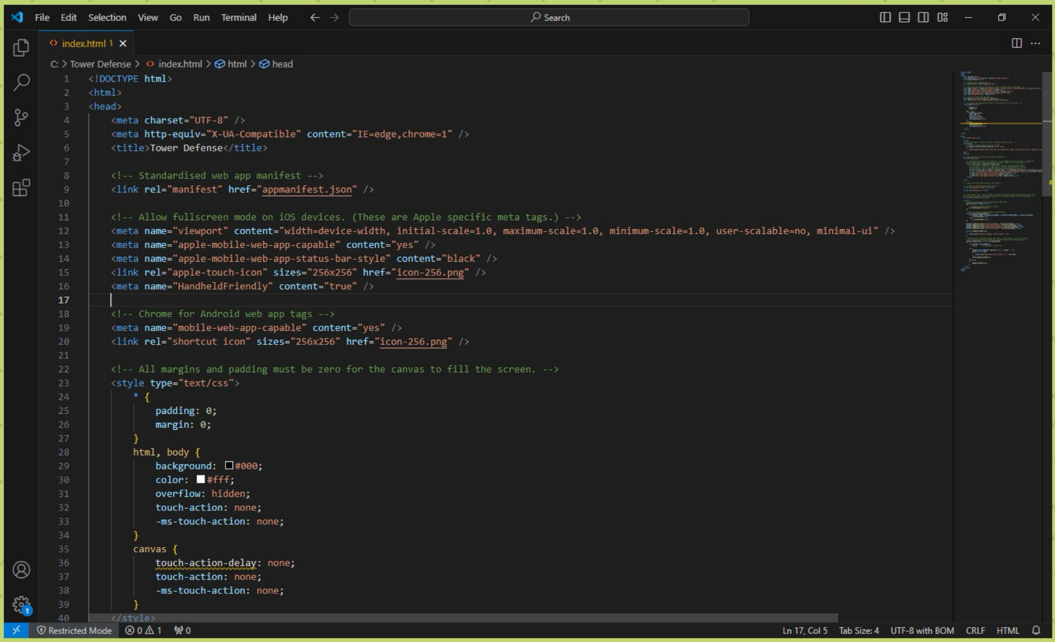
Dynamic Difficulty Scaling: As level progress, rabbit numbers increase, become more sophisticated and they employ more complex maneuvers to reach the cake.

Sprinkler Shooter: Fires confetti sprinkles with speed and kill the enemies.

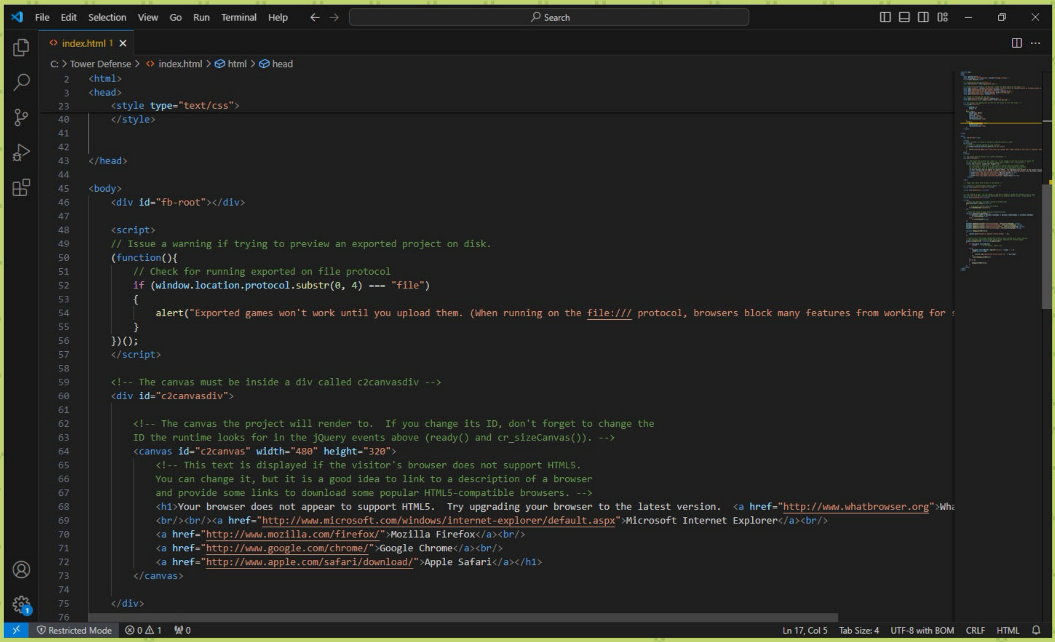
The gameplay follows a standard tower defense structure, with waves of enemies (rabbits) attacking the base (house with the cake). Players must strategically place turrets around the garden to fend off the rabbits.

After defeating all the rabbits, “I” can finally enjoy the birthday celebration. “I”'ll do a bit of parkour to light up the candle, and there will be a mini article “I” wrote for my girlfriend. Then, “I” join my girlfriend to sing the birthday song, completing the special day together.

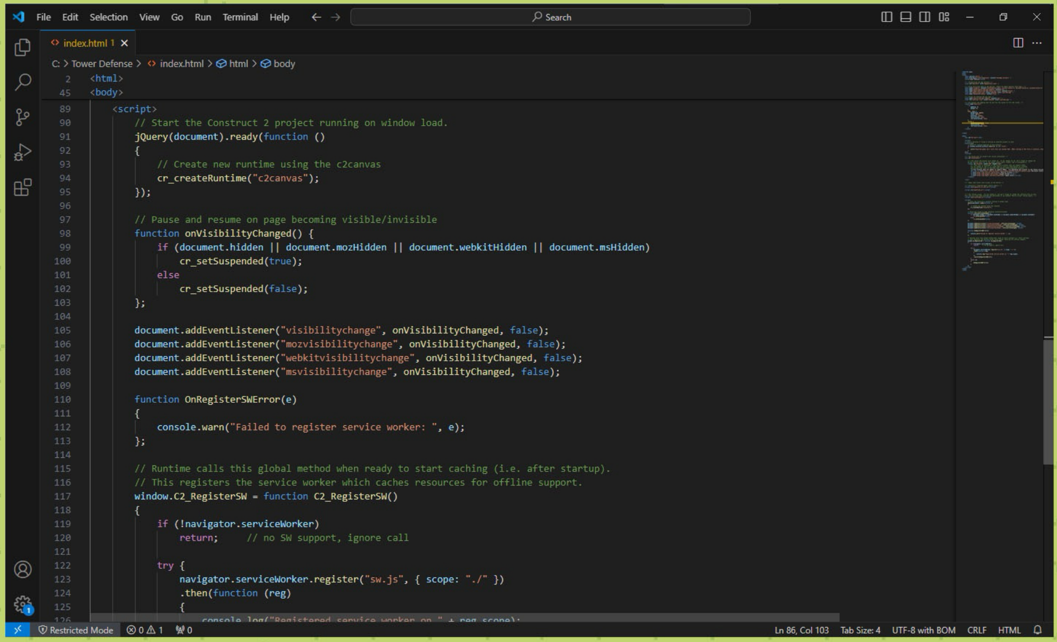




```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
    <title>Tower Defense</title>

    <!-- Standardised web app manifest -->
    <link rel="manifest" href="appmanifest.json" />

    <!-- Allow fullscreen mode on iOS devices. (These are Apple specific meta tags.) -->
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-scalable=no, minimal-ui" />
    <meta name="apple-mobile-web-app-capable" content="yes" />
    <meta name="apple-mobile-web-app-status-bar-style" content="black" />
    <link rel="apple-touch-icon" sizes="256x256" href="icon-256.png" />
    <meta name="HandheldFriendly" content="true" />

    <!-- Chrome for Android web app tags -->
    <meta name="mobile-web-app-capable" content="yes" />
    <link rel="shortcut icon" sizes="256x256" href="icon-256.png" />

    <!-- All margins and padding must be zero for the canvas to fill the screen. -->
    <style type="text/css">
        * {
            padding: 0;
            margin: 0;
        }
        html, body {
            background: #000;
            color: #fff;
            overflow: hidden;
            touch-action: none;
            -ms-touch-action: none;
        }
        canvas {
            touch-action-delay: none;
            touch-action: none;
            -ms-touch-action: none;
        }
    </style>
```



```html
<html>
<head>
    <style type="text/css">
    </style>


</head>

<body>
    <div id="fb-root"></div>

    <script>
    // Issue a warning if trying to preview an exported project on disk.
    (function(){
        // Check for running exported on file protocol
        if (window.location.protocol.substr(0, 4) === "file")
        {
            alert("Exported games won't work until you upload them. (When running on the file:/// protocol, browsers block many features from working for
        }
    })();
    </script>

    <!-- The canvas must be inside a div called c2canvasdiv -->
    <div id="c2canvasdiv">

        <!-- The canvas the project will render to.  If you change its ID, don't forget to change the
        ID the runtime looks for in the jQuery events above (ready() and cr_sizeCanvas()). -->
        <canvas id="c2canvas" width="480" height="320">
            <!-- This text is displayed if the visitor's browser does not support HTML5.
            You can change it, but it is a good idea to link to a description of a browser
            and provide some links to download some popular HTML5-compatible browsers. -->
            <h1>Your browser does not appear to support HTML5.  Try upgrading your browser to the latest version.  <a href="http://www.whatbrowser.org">Wha
            <br/><br/><a href="http://www.microsoft.com/windows/internet-explorer/default.aspx">Microsoft Internet Explorer</a><br/>
            <a href="http://www.mozilla.com/firefox/">Mozilla Firefox</a><br/>
            <a href="http://www.google.com/chrome/">Google Chrome</a><br/>
            <a href="http://www.apple.com/safari/download/">Apple Safari</a></h1>
        </canvas>

    </div>

```



```html
<html>
<body>
    <script>
        // Start the Construct 2 project running on window load.
        jQuery(document).ready(function ()
        {
            // Create new runtime using the c2canvas
            cr_createRuntime("c2canvas");
        });

        // Pause and resume on page becoming visible/invisible
        function onVisibilityChanged() {
            if (document.hidden || document.mozHidden || document.webkitHidden || document.msHidden)
                cr_setSuspended(true);
            else
                cr_setSuspended(false);
        };

        document.addEventListener("visibilitychange", onVisibilityChanged, false);
        document.addEventListener("mozvisibilitychange", onVisibilityChanged, false);
        document.addEventListener("webkitvisibilitychange", onVisibilityChanged, false);
        document.addEventListener("msvisibilitychange", onVisibilityChanged, false);

        function OnRegisterSWError(e)
        {
            console.warn("Failed to register service worker: ", e);
        };

        // Runtime calls this global method when ready to start caching (i.e. after startup).
        // This registers the service worker which caches resources for offline support.
        window.C2_RegisterSW = function C2_RegisterSW()
        {
            if (!navigator.serviceWorker)
                return;     // no SW support, ignore call

            try {
                navigator.serviceWorker.register("sw.js", { scope: "./" })
                .then(function (reg)
                {
                    console.log("Registered service worker on " + reg.scope);
```

This HTML file is designed as the setup for a web-based "Tower Defense" game. Here's a breakdown of each part:

<!DOCTYPE html>

Defines the document as an HTML5 file.

- *<html>, <head>, and <body> Tags*
- *<html>: The root element for the HTML document.*
- *<head>: Contains metadata and links for resources like stylesheets, scripts, and the page's title.*
- *<body>: Contains the visible content on the web page.*

Metadata Section in <head>

- *Character Set: <meta charset="UTF-8" /> defines the character encoding for the document.*
- *Compatibility Mode: <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" /> helps ensure compatibility with IE and Chrome.*
- *Title: <title>Tower Defense</title> sets the page title.*
- *Manifest: <link rel="manifest" href="appmanifest.json" /> links to the app manifest for progressive web app (PWA) capabilities.*

CSS <style> Block

The CSS ensures the canvas fills the screen without margins or padding:

- *Universal selector *: Resets padding and margins.*
- *html, body: Sets background and font colors, disables scrolling, and controls touch events.*
- *canvas: Sets the canvas to disable touch delay, optimizing responsiveness.*

JavaScript for Offline and Visibility Management

- *Disk Warning: A script checks if the file is being run locally (from the file:/// protocol) and alerts the user if so, since some features require it to be hosted online.*
- *Runtime Initialization: The game's runtime (c2runtime.js) is initialized, rendering the game within a <canvas> element.*
- *Visibility Events: Events to pause/resume the game based on page visibility, to conserve resources when the game is hidden.*
- *Service Worker Registration: Registers a service worker (sw.js) to enable offline support for the game, providing caching of resources for Progressive Web App (PWA) functionality.*

Canvas Setup in <body>

The main game interface is in a <canvas> element nested within a <div> with an ID of c2canvasdiv. If the browser doesn't support HTML5, a fallback message suggests browser options.

Scripts and Runtime

- *jQuery: A required library for the game.*
- *Pathfinding and Runtime Scripts: pathfind.js and c2runtime.js handle game logic and pathfinding.*
- *Service Worker and Runtime Event Handling: JavaScript is used to manage the service worker and handle caching, making the game more accessible offline.*

This setup is typical for web-based games and is optimized for both desktop and mobile browsers, with specific considerations for iOS and Android to ensure fullscreen, responsive, and offline capabilities.

Game Link: https://turretdefense.netlify.app