UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

PROJECT

TETRIS BROWSER GAME

MASTER

OF

COMPUTER APPLICATION

SUBMITTED BY

RITIK CHAUHAN

UID-24MCA20311

SUBMITTED TO

MR MANDEEP SINGH

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

CERTIFICATE

This is to certify that Ritik chauhan (UID- 24MCA20311) have successfully completed the project title "Introduction to Virtualization with VirtualBox in Linux" at University Institute of Computing under my supervision and guidance in the fulfilment of requirements of first semester, Master of Computer Application- Specialization in General. Of Chandigarh University, Mohali, Punjab.

Dr. Abdullah	Mr Mandeep Singh
Head of the Department	Project Guide Supervisor
UIC	UIC

DECLARATION

We hereby declare that the project entitled "Tetris Browser Game" submitted for the Master of computer Application Degree is our original work and the project

has not formed the basis or submitted for the award of any degree.

Ritik Chauhan

Signature:

Place

ACKNOWLEDGEMENT

We take this opportunity with much pleasure to thank all the people who have helped us through the course of our project development. We would like to express our sincere thanks to for his valuable guidance ,review and support in completing our project.

We are also thankful to our teachers Mandeep Singh for helping us in all possible ways. We would also like to express our gratitude to Dr. Abdullah (HOD, Master of computer Application) for his guidance. Your valuable guidance and suggestions helped us in various phase of the completion of this project. We will always be thankful to you in this regard.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

CONTENTS

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

INTRODUCTION

Background Tetris is a classic puzzle video game originally designed and programmed by Alexey Pajitnov in the Soviet Union. Released on June 6, 1984, the game has since become one of the most popular and iconic games of all time. Tetris involves players manipulating falling tetrominoes (geometric shapes composed of four square blocks each) with the aim of creating complete horizontal lines on the game board. Once a line is completed, it disappears, making room for new tetrominoes. The game ends when the tetrominoes stack up to the top of the screen.

Tetris has been widely recognized for its simple yet addictive gameplay, and it has been ported to a multitude of platforms, from early personal computers and game consoles to modern smartphones and web browsers. Its enduring popularity is a testament to its elegant design and the challenge it offers to players of all skill levels.

Objective

The objective of this project is to develop a browser-based version of Tetris. This involves creating a functional and engaging game that can be played directly within a web browser without the need for additional software. The project aims to replicate the classic Tetris experience while incorporating modern web development

techniques. The primary goals include designing an intuitive user interface, implementing core game mechanics such as tetromino movement and line clearing, and ensuring smooth performance across different browsers and devices.

By the end of this project, we hope to achieve a deeper understanding of web development, including HTML, CSS, and JavaScript, and gain practical experience in game design and programming. Additionally, this project will serve as a demonstration of our ability to apply theoretical knowledge to create a functional and interactive web application.

PROJECT PLANNING

Scope

The scope of this project is to develop a functional and engaging Tetris game that can be played directly within a web browser. The main features to be implemented include:

• A game board where tetrominoes fall and can be moved or rotated by the player.

• A scoring system that rewards players for clearing lines.

• Game controls for moving and rotating tetrominoes, as well as pausing and restarting the game.

• A user interface displaying the current score, next tetromino, and game status.

• Smooth performance across different browsers and devices.

Technologies Used

To achieve the project goals, the following technologies and tools were used:

• HTML: For structuring the game interface.

• CSS: For styling the game elements and ensuring a responsive design.

• JavaScript: For implementing the game logic and interactions.

DESIGN AND DEVELOPMENT

Game Design

Game Board: The game board is a 10x20 grid where tetrominoes fall and are manipulated by the player. This grid is implemented using HTML , which allows for efficient rendering of the game pieces and animations.

• Tetrominoes: The game features seven different tetromino shapes (I, O, T, S, Z, J, L). Each tetromino is composed of four blocks arranged in specific configurations. The shapes can be moved left, right, and down, and rotated clockwise or counterclockwise. Each shape is represented as a matrix of values indicating the presence of blocks

. • Scoring System: The scoring system awards points for each line cleared. Clearing one line scores a base amount of points, while clearing multiple lines at once awards higher points (e.g., clearing

four lines simultaneously results in a "Tetris" and the highest score).

This incentivizes players to strategize and aim for multiple line clears.

• **Game Controls**: Players interact with the game using keyboard controls: o Arrow keys to move tetrominoes left, right, and down.

Up arrow key or spacebar to rotate tetrominoes.

Pause button to pause the game.

Restart button to restart the game. The controls are designed to be intuitive and responsive to provide a smooth gameplay experience.

Development Process

The development process followed these steps:

1**. Initial Setup**: Setting up the project structure involved creating HTML, CSS, and JavaScript files. The HTML file includes the various element and UI components. The CSS styles the game and UI elements, ensuring a clean and responsive layout. The JavaScript contains the core game logic and interactions.

2. **Game Board Implementation**: The game board is implemented using the HTML Canvas API. The Canvas is initialized with a 10x20 grid, and the grid cells are drawn using JavaScript. Each cell can be empty or occupied by a block of a tetromino, and the state of the grid is continuously updated as the game progresses.

3. **Tetromino Generation**: Tetrominoes are generated randomly from the set of seven shapes. Each tetromino is represented as a matrix of blocks, and its position and rotation are tracked using JavaScript. The tetromino is rendered on the Canvas, and its position is updated based on player input and gravity (automatic downward movement).
4. **Movement and Rotation**: Keyboard event listeners are added to capture player inputs for moving and rotating tetrominoes. The game logic includes collision detection to ensure that tetrominoes do not move into occupied cells or outside the game board. The rotation logic also checks for valid rotations, adjusting the position if necessary to fit within the game board boundaries.

5. **Line Clearing**: The game checks for complete lines at each step. When a complete line is detected, it is cleared, and the lines above it are shifted down. This process is implemented using a loop that checks each row of the grid. The score is updated based on the number of lines cleared in a single move, and the cleared lines are visually removed from the Canvas.

6. U**ser Interface**: The user interface includes elements such as the current score display, the next tetromino preview, and game status indicators (e.g., "Game Over" message). These elements are updated in real-time based on the game state. CSS is used to style the UI elements, ensuring they are visually appealing and responsive.

7. **Testing and Optimization:** The game is tested across different browsers and devices to ensure compatibility and smooth performance. Testing involves playing the game, identifying bugs, and fixing them. Performance optimization includes reducing unnecessary computations, optimizing rendering techniques, and ensuring efficient use of JavaScript functions.

Challenges and Solutions

During the development process, several challenges were encountered:

• **Collision Detection**: Ensuring accurate collision detection for tetromino movement and rotation was challenging. This required careful implementation and testing of the collision detection logic, including edge cases where tetrominoes are near the game board boundaries or other tetrominoes.

• **Performance Optimization**: Optimizing the game for smooth performance across different browsers involved minimizing unnecessary computations and using efficient rendering techniques. This included reducing the frequency of Canvas redraws, optimizing the game loop, and ensuring that JavaScript functions executed efficiently.

USER INTERFACE

Design The user interface for the Tetris browser game is designed to be simple and intuitive. It includes the following elements:

• **Game Board**: The main area where the tetrominoes fall and are manipulated by the player.

• **Score Display**: Shows the current score, which updates in realtime as the player clears lines.

• **Next Tetromino Preview:** Displays the next tetromino that will appear, allowing the player to plan their moves.

• **Game Status Indicators**: Displays messages such as "Game Over" to inform the player of the current game state.

• **Control Buttons: Buttons** for starting, pausing, and restarting the game

User Experience

The user experience is designed to be smooth and enjoyable:

• **Responsive Controls:** The game responds quickly to player inputs, ensuring that movements and rotations of tetrominoes feel natural and responsive.

• **Clear Visuals**: The game board and tetrominoes are clearly rendered, with distinct colours for each tetromino shape to help players quickly identify them.

• **Feedback**: Visual and audio feedback is provided for important actions, such as clearing lines or reaching a game over state. This enhances the player's engagement with the game.

FEATURES

The Tetris browser game project includes several key features that enhance gameplay and user experience:

The Tetris browser game project includes several key features that enhance gameplay and user experience:

1. **Interactive Game Board:** Grid Layout: A 10x20 grid serves as the game board where tetrominoes fall and are manipulated by the player.

Canvas Rendering: Utilizes HTML5 Canvas for efficient rendering of game elements, ensuring smooth animations and responsive gameplay.

2. **Tetromino Shapes**: Seven Shapes: Includes all classic Tetris shapes (I, O, T, S, Z, J, L), each composed of four blocks. Random Generation: Tetrominoes are generated randomly, providing dynamic gameplay and increasing replayability.

3. **Gameplay Mechanics**: Movement and Rotation: Players can move tetrominoes left, right, and down using arrow keys or similar controls. They can also rotate tetrominoes clockwise to fit into gaps.

Collision Detection: Ensures tetrominoes do not overlap or extend beyond the game board boundaries, enhancing gameplay accuracy.

Line Clearing: Completed horizontal lines of blocks are cleared from the board, earning points for the player. Clearing multiple lines simultaneously rewards bonus points.

4. **User Interface (UI**): Score Display: Real-time update of the player's score based on cleared lines. 16

Next Tetromino Preview: Shows the upcoming tetromino shape, helping players plan their moves strategically.

Game Status Indicators: Displays messages such as "Game Over" or "Paused" to inform players of their current game status. o Control Buttons: Includes intuitive buttons for starting, pausing, and restarting the game, enhancing user interaction.

5. **Responsive Design**: Cross-Device Compatibility: Designed to be playable on various devices and screen sizes, ensuring a consistent user experience from desktops to mobile devices.

CSS Styling: Uses responsive design techniques in CSS to adapt game elements and UI components for optimal display on different screens.

6. **Performance Optimization:** Smooth Animation: Utilizes various functions for efficient frame rendering, ensuring smooth animation and reducing CPU usage.

Code Efficiency: Optimizes JavaScript code for game logic and rendering, minimizing computational overhead and enhancing overall performance.

7. **Accessibility and Usability**: Clear Visual Feedback: Provides clear visual cues and feedback for player actions, enhancing usability for both novice and experienced players.

Keyboard Controls: Supports keyboard input for intuitive gameplay, ensuring accessibility for players without requiring additional hardware or controls. 17

8**. Music playback support**: Music playback: When you load up the game and press any button, the music will start playing

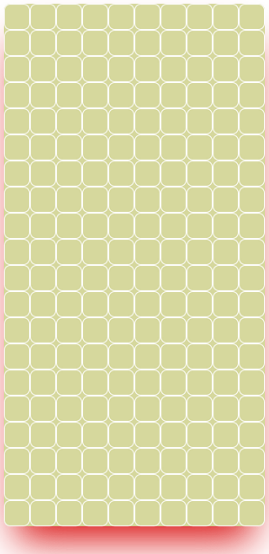
Music control: You can mute the background music using the dedicated button or use the volume slider to adjust the level of music.

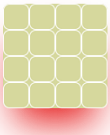
SNAPSHOTS

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Score: 0

Start/Pause Restart

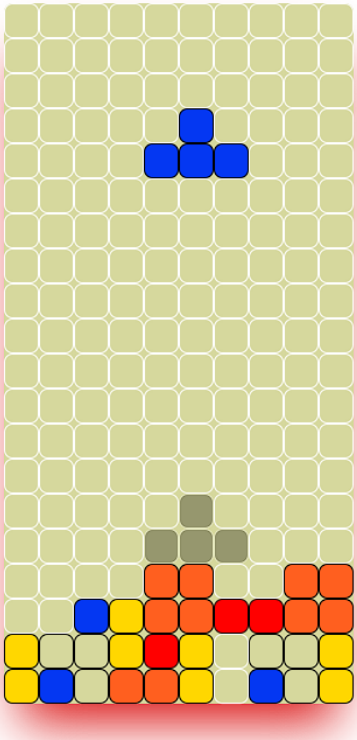


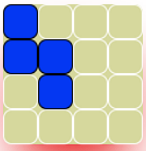


Show/Hide Virtual Controls

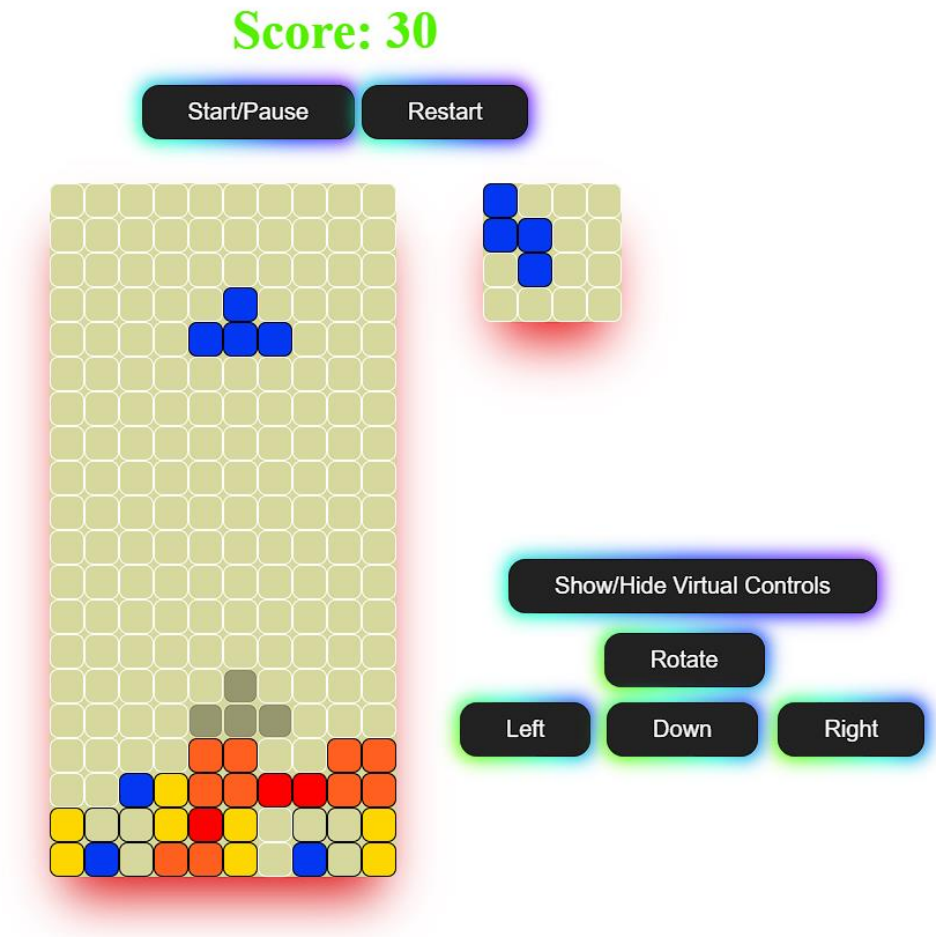
Score: 30

Start/Pause Restart





Show/Hide Virtual Controls



FUNCTIONS

Draw Function

• **Purpose**: The draw function is responsible for rendering the Tetris piece on the grid.

• **Functionality:** o It iterates over the blocks of the current Tetris piece.

For each block, it determines its position on the grid.

It then updates the corresponding cell in the HTML table to reflect the block's presence, usually by adding a class that styles the cell with the piece's color.

Undraw Function

• **Purpose**: The undraw function removes the visual representation of the Tetris piece from the grid

. • **Functionality**: o

Similar to the draw function, it iterates over the blocks of the current Tetris piece.

For each block, it determines its position on the grid.

It then resets the corresponding cell in the HTML table to its default state, usually by removing the class that styles the cell.

These functions work together to update the game display as pieces move, rotate, or are placed on the grid. By undrawing a piece before moving it and drawing it again in the new position, the game ensures that the visual representation is accurate and current

MoveRight Function

• **Purpose:** The moveRight function moves the current Tetris piece one step to the right.

• **Functionality**: It first uses the undraw function to clear the piece from its current position.

It checks if moving the piece to the right is valid (i.e., it doesn't collide with other pieces or the edge of the grid).

If the move is valid, it updates the piece's position by incrementing the x-coordinate of each block.

It then calls the draw function to render the piece in its new position.

MoveLeft Function

• **Purpose:** The moveLeft function moves the current Tetris piece one step to the left.

• **Functionality:** It uses the undraw function to clear the piece from its current position.

It checks if moving the piece to the left is valid.

If the move is valid, it updates the piece's position by decrementing the x-coordinate of each block.

It then calls the draw function to render the piece in its new position

MoveDown Function

• **Purpose**: The moveDown function moves the current Tetris piece one step down.

• **Functionality**:

It uses the undraw function to clear the piece from its current position.

It checks if moving the piece down is valid.

If the move is valid, it updates the piece's position by incrementing the y-coordinate of each block.

It then calls the draw function to render the piece in its new position. 40

If the move down is not valid, it means the piece has landed, and the piece is added to the grid permanently. A new piece is then spawned.

Rotate Function

• **Purpose:** The rotate function rotates the current Tetris piece.

• **Functionality:** It uses the undraw function to clear the piece from its current position.

It determines the new orientation of the piece by updating the coordinates of its blocks.

It checks if the new orientation is valid.

If the rotation is valid, it updates the piece's blocks to the new coordinates.

It then calls the draw function to render the piece in its new orientation.

Freeze Function

• **Purpose**: The freeze function is called when a Tetris piece lands and can no longer move downward.

• **Functionality**: It iterates through each block of the current piece.

For each block, it updates the grid to mark the position of the block as occupied.

It checks for complete lines in the grid. If a complete line is found, it is cleared, and the above lines are shifted down.

Restart Function

• Purpose: The restart function is used to reset the game to its initial state, allowing the player to start a new game.

• **Functionality**: It clears the game grid.

It resets the score and other game-related variables.

It spawns a new piece to start the game anew.

It ensures that the game-over flag is cleared so the game can proceed normally

Game over Function

• **Purpose:** The gameOver function is called when the game detects that there is no space to spawn a new Tetris piece.

• **Functionality:** It sets a flag indicating that the game is over.

It might display a "Game Over" message to the player.

It typically disables further game actions, such as moving pieces.

Pause Function

• **Purpose**: The pause function temporarily halts the game, allowing the player to take a break.

• **Functionality:** It sets a flag indicating that the game is paused.

It stops the automatic falling of Tetris pieces.

It might display a "Paused" message to inform the player.

It ensures that no game actions can be performed while the game is paused.

addScore Function

• **Purpose:** The addScore function updates the player's score based on the number of lines cleared.

• **Functionality:** It takes the number of lines cleared as an input.

It calculates the score increment based on the number of lines. Typically, clearing more lines at once results in a higher score increment.

It updates the total score with the calculated increment.

It might update the display to show the new score to the player.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

CSS

Explanation of Glowing RGB Effect on Buttons

1. **Initial Styles**: The buttons are styled with basic visual properties like padding, border-radius, background color, text color, and outline. A box-shadow property is applied to create the illusion of a glow around the button
2. . **2. Keyframes for Animation**: @keyframes are used to define the color changes for the glow effect. This involves setting different colors at various percentages of the animation's duration.

Colors transition smoothly between red, green, and blue to create a dynamic RGB effect.

Explanation of "?" Icon and Hover Effect1.

Icon Usage:

The "?" icon is typically used to indicate help or additional information. In the context of your Tetris game, it might provide tips, instructions, or game controls when clicked or hovered over. 2.

Hover Effect:

CSS Styling: Initially, the "?" icon is styled with basic properties like size, color, and positioning.

Hover State: When the user hovers over the "?" icon, additional styling properties are applied using the :hover pseudo-class.

Transition Effects: CSS transition properties can be used to animate the change in appearance when the user hovers over the icon, making the interaction smoother and more intuitive.

Tooltip or Modal: Clicking or hovering over the "?" icon might trigger a tooltip or modal box to appear, containing relevant information about the game controls, scoring, or gameplay tips.

3. **Functionality:**

   Click Action: Clicking the "?" icon could toggle the display of a tooltip or modal with helpful information.

   Hover Action: Hovering over the "?" icon could change its appearance slightly (e.g., increase opacity, change color) to indicate it's interactive

. By implementing these elements effectively, we can enhance user experience by providing clear guidance and information about our Tetris game without cluttering the main gameplay interface.

CONCLUSION

The development of the Tetris browser game has been a comprehensive and enriching experience, blending fundamental concepts in game design, web development, and software engineering. This project demonstrates how a classic game like Tetris can be effectively implemented using modern web technologies, providing both a nostalgic and educational journey.

Learning Outcomes

This project provided valuable insights and practical experience in several areas:

1. **Web Development**: Building the game deepened understanding of HTML, CSS, and JavaScript, particularly in the context of interactive and dynamic web applications.

2. **Game Design**: Implementing Tetris highlighted the importance of game **mechanics,** user feedback, and balancing challenge with playability. It also underscored the need for clear and responsive controls to enhance the player experience. 51

3**. Problem-Solving**: The project fostered problem-solving skills, requiring innovative solutions to technical challenges such as collision detection, performance optimization, and responsive design.