Game Title: Pixel Painter Showdown

Description: Pixel Painter Showdown is a multiplayer competitive game where players take on the role of pixel artists and compete to create the most impressive artwork within a limited time frame. The catch is that they can only use a limited number of pixels to craft their masterpiece.

Basic game

So, I will explain here's a step-by-step explanation of how the "Pixel Painter Showdown" game is executed:

1. Starting the Game:

- Players run the compiled executable (e.g., pixel painter) in their terminal or command prompt to start the game.

- The game greets the players and presents them with options to create a new room or join an existing room in the lobby.

2. Lobby and Room Setup:

- Players can choose to join an existing room by selecting its name from the list of available rooms or create a new room with a custom name.

- Once enough players have joined a room (e.g., at least two players), the game announces the start of the round and proceeds to the next phase.

3. Canvas and Colors

- Each player is presented with a blank canvas represented as a grid of pixels, along with a menu of available colors they can use to draw.

- Players take turns choosing colors from the menu, one at a time, until all players have selected their desired colors.

4. Limited Pixels and Time Limit:

- The game informs the players about the number of pixels they must create their artwork (e.g., 100 pixels).

- It sets a time limit for the round (e.g., 5 minutes) and displays a countdown timer for players to keep track of the remaining time.

5. Pixel Art Creation:

- Players take turns placing pixels on the canvas using their selected colors.

- They use the arrow keys or WASD to navigate the cursor on the canvas and press 'Enter' or 'Space' to place a pixel at the cursor's position.

- The game enforces the restriction on the number of pixels each player can use, so they need to strategize and plan their artwork carefully.

6. Round End and Voting:

- After the time limit is up, the game stops players from placing any more pixels on the canvas.

- All the artworks from different players are displayed anonymously on the screen.

- Players can review and appreciate each other's creations but cannot make any changes to the canvas at this point.

7. Winner Announcement

- Players are prompted to vote for the artwork they find most impressive. They might vote by entering the number or name associated with the artwork they choose.

- The game counts the votes, determines the artwork with the most votes, and announces the winner of the round.

8. Pixel Rewards:

- The winning artist is rewarded with extra pixels that they can use in the next round, allowing them to expand their creativity further.

9. Pixel Shop (Optional)

- Between rounds, players can visit a "Pixel Shop" where they can spend their earned pixels to unlock new colors, patterns, or special effects to use in future rounds, enhancing their artistic options.

10. Game Over

- The game continues with new rounds if players choose to play.

- Optionally, the game may have an end condition, like a predetermined number of rounds or a set time for the overall game.

Once the end condition is met, the game is over, and the final leaderboard is displayed, showing each player's wins and total pixel count.

This execution flow provides players with an engaging and creative experience as they compete to create the best artwork within the given constraints of limited pixels and time. The game fosters friendly competition and encourages players to continuously improve their pixel art skills.

Lobby and room setup

To create a simple lobby system where players can join or create rooms to play the "Pixel Painter Showdown" game, we can use basic user input and data structures to keep track of the players and their respective rooms.

Here's a simplified explanation of how the room/lobby system works in the code:

1. Structures:

The code uses two structures: Player and Room- The Player structure stores the name of the player and the room number they are currently in.

- The Room structure represents a room and contains an array of players.

2. create **Room** Function:

- This function allows a player to create a new room and join it.

- It takes the array of rooms, the number of existing rooms, and a pointer to the player as arguments.

- It checks if the maximum number of rooms has been reached.

- If not, it creates a new room, assigns a unique ID to the room, and adds the player to it.

- The player's room number is updated, and the number of rooms is increased.

3. **join Room** Function

- This function allows a player to join an existing room.

- It takes the array of rooms, the number of existing rooms, the player, and the room ID to join as arguments.

- It checks if the specified room ID is valid and if the room has available slots for players.

- If so, it adds the player to the room, updates the player's room number, and increments the number of players in the room.

4. **Main** Function

- The main function initializes arrays to store players and rooms and keeps track of the number of existing players and rooms.

- It takes the player's name as input to create a new player.

- The main menu is displayed with options to create a new room, join an existing room, start the game, or quit.

- Depending on the user's choice, the corresponding functions (**createRoom** and **joinRoom**) are called to handle room creation and joining.

- When the user chooses to start the game, the code proceeds to the game logic (not shown in the provided code) using the player's room number to identify the game room.

Canvas and colors

In the context of the "Pixel Painter Showdown" game, the canvas and colors play essential roles in facilitating the creation of pixel art by the players.

1. Canvas

- The canvas represents the drawing area where players can place their pixels to create their artwork.

- In this game, the canvas is often represented as a two-dimensional grid of cells, where each cell corresponds to a pixel that can be filled with a color.

- The size of the canvas is defined by the constant **CANVAS_SIZE** in the code. For example, CANVAS_SIZE could be set to 10, creating a 10x10 grid.

2 Colors

- Colors represent the different options available to players for filling the pixels on the canvas. Players can choose from a limited set of colors to create their artwork.

- In the code example provided earlier, the available colors are represented as characters in the colors array.

- The available colors are defined by the constant **MAX_COLORS** in the code. For example, **MAX_COLORS** could be set to 5, providing players with five color options (e.g., red, green, blue, yellow, white).

- Players select their desired color using a number corresponding to the color's position in the **colors** array.

In the code example, the colors are represented by single uppercase letters:

- 'R' for red

- 'G' for green

- 'B' for blue

- 'Y' for yellow

- 'W' for white

However, you can customize the available colors to your liking by adding or modifying entries in the colors array.

Pixel art creation

Canvas and Colors Selection:

At the beginning of each player's turn, the game displays the canvas grid, which represents the drawing area.

Players are prompted to choose a color from the available options. The available colors are listed with corresponding numbers (e.g., 1 for red, 2 for green, etc.).

Players enter the number of the color they want to use for their current turn.

Drawing on the Canvas:

After selecting a color, players can start drawing on the canvas.

The game prompts the player to enter the coordinates (x, y) of the pixel they want to fill with the chosen color.

Players input the coordinates using the keyboard, and the game validates if the input is within the canvas boundaries.

If the coordinates are valid (within the canvas boundaries) and the chosen pixel is not already filled with a color, the game fills the selected pixel with the chosen color.

The player's remaining pixels are decreased by one after successfully placing a pixel.

Time Limit

Setting the Time Limit:

The time limit is a predefined duration that is set by the game before each round starts. For example, the time limit could be 5 minutes (300 seconds) for each round.

The game indicates the time limit to the players at the beginning of the round, so they are aware of the remaining time as they create their artwork.

Managing Time During the Round:

The game uses a timer to keep track of the elapsed time during the round. The timer starts counting down as soon as the round begins.

As players take their turns to create pixel art, the game continuously updates and displays the remaining time on the screen to provide real-time feedback.

Time Constraints and Strategy:

The time limit imposes constraints on the players' creative process. Players must efficiently use their time to create their artwork, making strategic decisions on where to place pixels and which details to include.

The limited time encourages players to think quickly and prioritize their creative choices, adding an exciting and challenging aspect to the game.

Round End:

When the time limit is reached, the round automatically ends, and players can no longer place pixels on the canvas.

The game transitions to the voting phase, where players review and vote on each other's artwork.

Handling Time Expired Players:

If a player uses up all their allotted time before the round ends, they can no longer place pixels on the canvas for that round. The game ensures that the turn ends for the player when they run out of time.

The player's artwork, as it stands at the end of their turn, is considered their final submission for the round.

Advantages of Time Limit:

The time limit adds excitement to the game, as players race against the clock to create their artwork before time runs out.

It prevents overly lengthy turns, ensuring that the game progresses at a steady pace.

The time limit encourages players to stay engaged and actively participate throughout the round**.**

Round ending and voting

Round Ending Phase:

The round-ending phase is triggered when the time limit for the current round expires or when all players have used up their allotted pixels.

At this point, players can no longer place pixels on the canvas, and the canvas displays the final artwork of each player as it stands.

Reviewing Artwork:

During the round-ending phase, all the completed artwork is displayed on the screen. Players can review and appreciate each other's creations.

The game ensures that the artwork is displayed anonymously, meaning that the player who created each artwork is not revealed until after the voting process.

Voting Process:

After the round-ending phase, the game proceeds to the **voting process**.

Each player is given the opportunity to vote for the artwork they find most impressive. This voting process can be simulated for demonstration purposes or implemented in various ways, depending on the game's requirements.

Winner announcement

Tallying Votes:

After the voting phase is complete, the game tallies the votes for each artwork. The number of votes received by each artwork is recorded.

Determining the Winner:

The artwork with the most votes is declared the winner of the round.

If there is a tie in the number of votes between two or more artworks, the game may implement tiebreaker rules to determine the winner. For example, the tiebreaker could be a random selection of one of the tied

artworks as the winner, or it could be a shared victory among the tied players.

Announcing the Winner:

Once the winner of the round is determined, the game announces the winner to all players.

The announcement can be displayed on the screen along with the name of the winning player or artwork number.

The winning artwork may also be highlighted or showcased on the screen to give it special recognition.

Round Winner Rewards:

The winning player receives recognition as the round winner, which can add a sense of accomplishment and motivation for the next rounds.

Optionally, the game may reward the winner with extra pixels that they can use in the next round. This reward can be a way to provide a slight advantage to the winner and encourage continued creativity.

Transition to the Next Round:

After the winner is announced and any rewards are given, the game proceeds to the next round, and players continue with their pixel art creation process.

If there are more rounds to play, the game loops back to the color selection and canvas drawing phases for the next round.

Final Winner Announcement:

If the game has a predetermined number of rounds, the winner of the entire game is determined based on the players' performance throughout all the rounds.

At the end of the final round, the game calculates the overall winner by considering factors such as the number of round wins and total votes received.

The final winner is announced, and the game displays the final leaderboard to showcase the ranking of all players based on their performance.

Game over

Determine the Final Winner:

After all the rounds have been played, the game calculates the final winner based on the players' performance throughout the game. This calculation may consider factors such as the total number of round wins, the total votes received, and any other game-specific metrics that contribute to determining the overall winner.

Announce the Final Winner:

The game announces the final winner to all players. The announcement can be displayed on the screen along with the name or identifier of the winning player or artwork.

The final winner receives special recognition and celebration for their outstanding performance throughout the game.

Display Final Leaderboard:

Along with announcing the final winner, the game displays the final leaderboard. The leaderboard shows the ranking of all players based on their performance in the game.

The leaderboard may include information such as the number of round wins, total votes received, and any other relevant metrics for each player.

Closing Remarks:

After announcing the final winner and displaying the leaderboard, the game may provide closing remarks or a summary of the game's outcome.

Players may be congratulated for their participation, and the winning player may receive special accolades for their exceptional artwork and creativity.

End of the Game:

The game officially ends after the game over phase.

Players have the option to play again or return to the main menu, depending on the game's design.

Optional Features:

In addition to declaring the final winner, the game over phase may also include additional statistics, such as the average number of votes

received per round for each player or the most popular artwork based on votes.

The game may provide the option to save or share the winning artwork or the final leaderboard for bragging rights or future reference.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define CANVAS_SIZE 10   // Size of the canvas grid (e.g., 10x10)
#define MAX_PIXELS 100   // Maximum number of pixels each player can use
#define MAX_COLORS 5     // Maximum number of available colors

// Function to initialize the canvas with blank spaces
void initializeCanvas(char canvas[][CANVAS_SIZE]) {
    for (int i = 0; i < CANVAS_SIZE; i++) {
        for (int j = 0; j < CANVAS_SIZE; j++) {
            canvas[i][j] = ' ';
        }
    }
}

// Function to display the canvas with the current artwork
void displayCanvas(char canvas[][CANVAS_SIZE]) {
    for (int i = 0; i < CANVAS_SIZE; i++) {
        for (int j = 0; j < CANVAS_SIZE; j++) {
```

```
            printf("%c ", canvas[i][j]);
        }
        printf("\n");
    }
}

// Function to handle player's turn to draw
void playerTurn(char canvas[][CANVAS_SIZE], char colors[], int playerPixels) {
    int x, y;
    char playerColor;
    int chosenColor;

    printf("Choose your color:\n");
    for (int i = 0; i < MAX_COLORS; i++) {
        printf("%d - %c\n", i + 1, colors[i]);
    }
    scanf("%d", &chosenColor);

    // Assign the chosen color to the player
    playerColor = colors[chosenColor - 1];
```

```
    while (playerPixels > 0) {
        printf("Remaining pixels: %d\n", playerPixels);
        displayCanvas(canvas);

        // Get player input for the position to place a pixel
        printf("Enter x and y coordinates to place a pixel: ");
        scanf("%d %d", &x, &y);

        // Check if the position is valid and place the pixel
        if (x >= 0 && x < CANVAS_SIZE && y >= 0 && y < CANVAS_SIZE) {
            if (canvas[x][y] == ' ') {
                canvas[x][y] = playerColor;
                playerPixels--;
            } else {
                printf("Pixel already placed there!\n");
            }
        } else {
            printf("Invalid coordinates! Try again.\n");
        }
    }
}
```

```
// Function to simulate the voting process and determine the winner
int determineWinner(int votes[], int numPlayers) {
    int maxVotes = 0;
    int winnerIndex = -1;

    for (int i = 0; i < numPlayers; i++) {
        if (votes[i] > maxVotes) {
            maxVotes = votes[i];
            winnerIndex = i;
        }
    }

    return winnerIndex;
}

int main() {
    char canvas[CANVAS_SIZE][CANVAS_SIZE];  // 2D array to store the canvas
    char colors[MAX_COLORS] = {'R', 'G', 'B', 'Y', 'W'};  // Available colors

    int numPlayers;  // Number of players (set based on lobby/room setup)
```

```
    int roundTime = 5 * 60;  // Time limit for each round in seconds (e.g., 5 minutes)

    int playerPixels[MAX_PLAYERS];  // Remaining pixels for each player
    int votes[MAX_PLAYERS] = {0};   // Array to store the votes for each player

    srand(time(0));  // Seed the random number generator with the current time

    // Game setup and lobby code goes here (e.g., room creation/joining)

    // Round starts
    for (int round = 1; round <= MAX_ROUNDS; round++) {
        // Initialize the canvas with blank spaces
        initializeCanvas(canvas);

        // Players choose colors one by one
        for (int i = 0; i < numPlayers; i++) {
            printf("Round %d, Player %d's turn:\n", round, i + 1);
            playerTurn(canvas, colors, MAX_PIXELS);
```

```
        // Save the remaining pixels for the current player for future
rounds
        playerPixels[i] = MAX_PIXELS;
    }

    // End of the round, display artworks and handle voting

    // Simulate voting process (for demonstration purposes)
    for (int i = 0; i < numPlayers; i++) {
        int vote;
        printf("Player %d, vote for the best artwork (enter player
number): ", i + 1);
        scanf("%d", &vote);

        // Ensure the vote is valid
        while (vote < 1 || vote > numPlayers || vote == i + 1) {
            printf("Invalid vote! Try again: ");
            scanf("%d", &vote);
        }

        votes[vote - 1]++;
    }
```

```
        // Determine the winner of the round
        int roundWinner = determineWinner(votes, numPlayers);
        printf("Round %d winner: Player %d\n", round, roundWinner + 1);
        // Reward the winning player with extra pixels (optional)

        // Clear votes for the next round
        for (int i = 0; i < numPlayers; i++) {
            votes[i] = 0;
        }
    }

    // End of the game, display final leaderboard and game-over
message

    return 0;
}
```