# Custom Project

Ruby2D is a versatile library for creating 2D games and applications in Ruby. This guide will walk you through building a simple chess game using Ruby2D, focusing on core features such as board setup, piece movement, and rendering.

## Setting up the evironment

Gem install ruby2d

## Structuring the Chess Game

You can create the nescceary file for the game following this:

Pieces/

bb.jpg

bb.png

bk.jpg

bn.jpg

bq.jpg

br.jpg

wb.jpg

wk.jpg

wk.png

wn.jpg

wn.png

wp.png

wq.jpg

wr.jpg

wr.png

main.rb

piece.rb

## Defining the Board and Pieces

To represent the board, you can implement bitboard approach. Firstly, you will need assign value for each piece and its color.

A screenshot of a computer

Description automatically generated

Then you can create an 2d array to store all of the piece position

A screen shot of a computer code

Description automatically generated

In the piece.rb file, you can create these method to represent each piece and its color, because we can represent its type by using it first 3-bit, we can use bitwise & to get the type, thus we get the type of each piece. For the color, as you can see it is define on the 5th bit and 4th bit, if the 4th bit is equal to 8, that means our piece color is white.

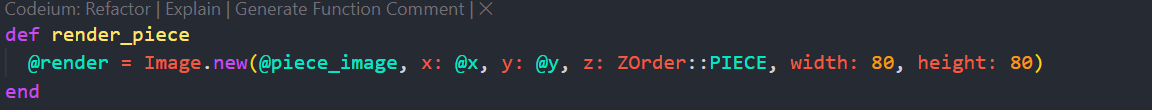
A screenshot of a computer program

Description automatically generated

To get the image location of each pieces, we can use this implementation for each type of pieceA screen shot of a computer code

Description automatically generated

To render a piece to the board, we can use the image creation function in ruby2d,



## Drawing the board

The draw\_board method creates the chessboard and displays the pieces. It loops through each square on the 8x8 grid, calculating the color based on the position to create the classic light and dark pattern. Dark squares are colored #6e4e36 and light squares #b99b75. Each square is drawn using the Square class and added to the @squares array. The method then checks if a piece is on each square by looking at the @board array. If a piece is found, it uses the piece\_image method to get the image, creates a Piece object, and renders it at the correct position. This ensures the board and pieces are displayed accurately and consistently.

A screen shot of a computer program

Description automatically generated

## Select a piece

The handle\_mouse\_click method processes mouse input when interacting with the chessboard. It first converts the mouse's x and y coordinates into corresponding board positions (rank and file) by dividing them by 80, which represents the size of each square on the board. This ensures accurate identification of the clicked square.

The method checks if a piece has already been clicked. If no piece is selected = it calls select\_piece to highlight and activate the piece at the clicked position. If a piece is already selected, it tries to move the piece or capture an opponent's piece by calling move\_piece\_or\_capture. This structure ensures smooth, logical handling of player interactions during the game, supporting both selecting and moving pieces while managing invalid actions gracefully.

A screen shot of a computer program

Description automatically generated

The select\_piece method handles the logic for selecting a piece on the board when the user clicks a square. It identifies the piece at the clicked position, generates valid moves, and visually updates the board to indicate the selected piece and its potential moves.

A computer screen shot of a code

Description automatically generated

## Moving or capturing piece

The move\_piece\_or\_capture method handles both the movement of a selected piece to a new square and the capturing of an opponent's piece. It ensures that a move is legal, updates the board, and deals with capturing logic.

A screen shot of a computer program

Description automatically generated

In ruby2d, we must manually remove or add a new image to the window so we will create a render at new position function that will remove the old image, and render the new updated piece position image. The move\_piece and render\_at\_new\_pos methods work together to handle the process of moving a chess piece from one square to another on the board. These methods update the piece's position, visually represent the movement, and ensure that the game board is updated accordingly. The capture piece function only need to remove that piece off the board.

A screenshot of a computer program

Description automatically generated

## Generating moves

A computer screen shot of text

Description automatically generated

To generating moves for each piece, we can create a simple script that iterating the possible directions add check if that moves is within the range of the board.

A computer screen shot of a program

Description automatically generated with medium confidence

If that move is legal, it will add to the moves array of each piece and when the user click on that piece, its legal moves will be draw on the board.

Demo of the game

A screenshot of a game

Description automatically generated

## References

[YouTube](https://www.youtube.com/watch?v=U4ogK0MIzqk)