CI401

Vickshan Vicknakumaran

Coursework: Simple Application

Breakout Game

Contents

[Description about the project and the aims 2](#_Toc71879901)

[Features 2](#_Toc71879902)

[Add rows of Bricks 2](#_Toc71879903)

[Add Play and Pause function 2](#_Toc71879904)

# Description about the project and the aims

The project which I had chosen was the Breakout game. The simple principle of the game is to destroy all rows of bricks by bouncing the ball using the bricks. When I had first ran the project, the things that were running were the scoreboard, ball, and the bat. Based on what I had saw, I decided to create the following aims:

* Add rows of bricks for the ball to hit and earn points
* Fix the bat from going off the window size.
* Add Pause and Continue function for the game

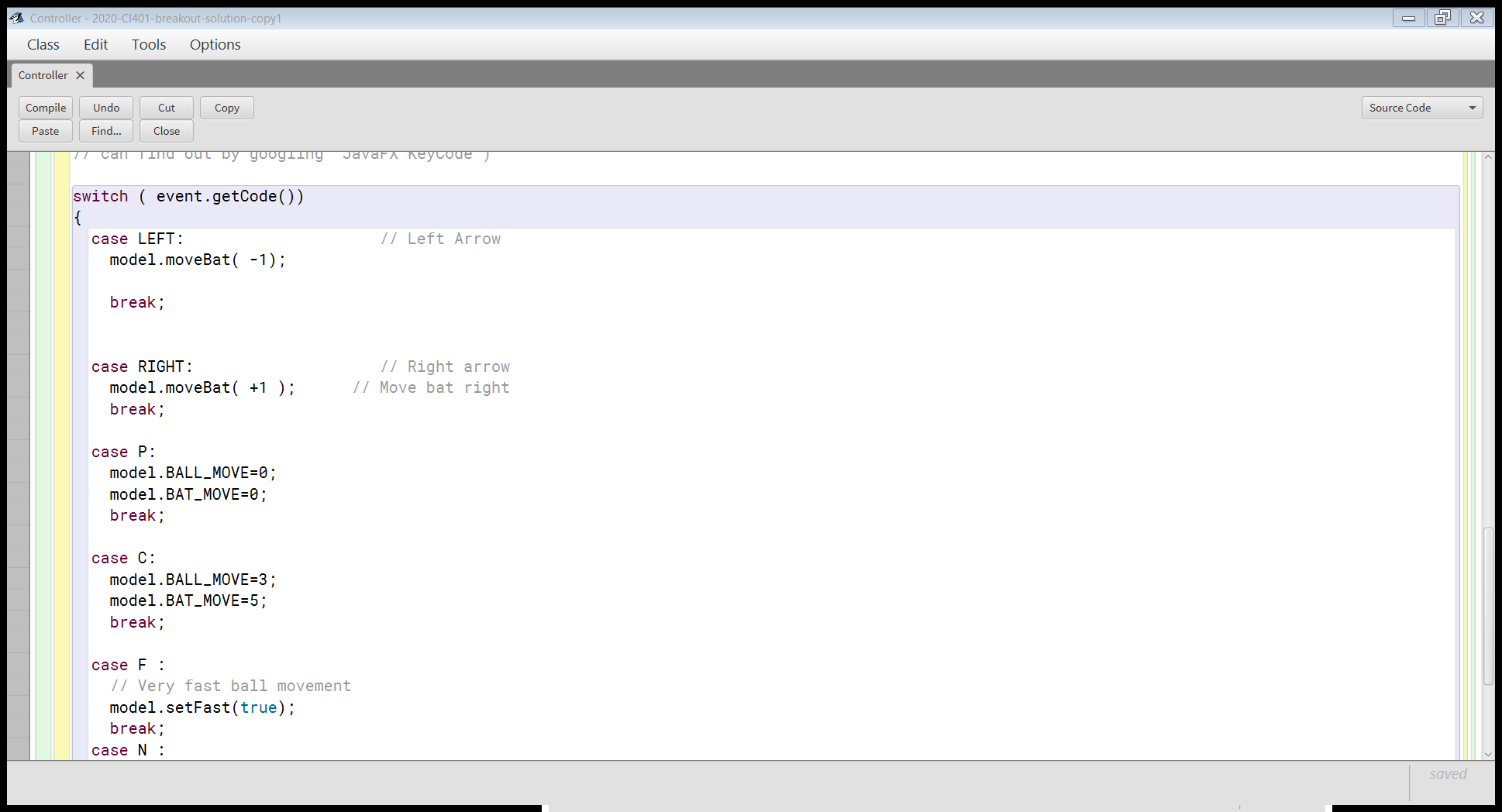
# Features

## Add rows of Bricks

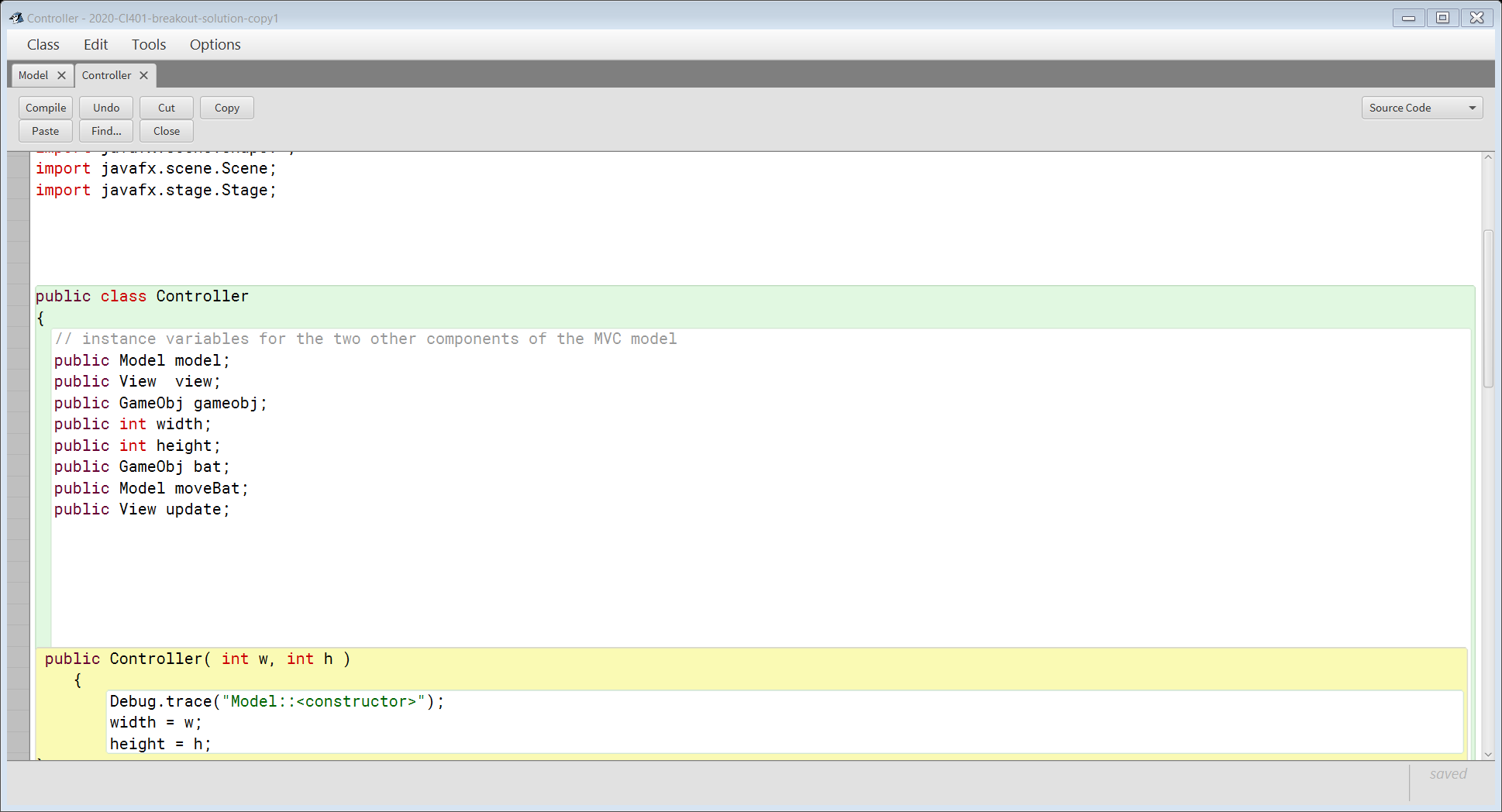


This code is in the Model. Firstly, I had created two variables called rows and column. The rows variable would specify the number of rows of bricks that I wish to add. The column variable specifies the number of bricks which will consist in one row. Since each row will have the same structure, I had decided to create a for loop which will create each row. As displayed in the screenshot above, I have stated a starting value “g” to 0 in which the loop will start and compared whether the number of rows I had wished is less than the current number of loops. In this we can consider that row acts as the limit for the number of times for statement loops. Every time it loops the variable” g” will add 1 and state the current number of repeated loops. For the number of columns, I had included another for loop inside the current loop as the columns are also in need to repeat so it completes the row. In this for statement I had used the same principle as the first for loop except the variable starting point is compared to the number columns that I wish to have. Under the following statement for each column to be drawn. When I had decided to add rows and columns of bricks, I had wanted each row to have different colours so that players could see each row easily. For me to achieve the aim, I had created an array which held several types of colours. Using this array, I had included inside for loop so that for each loop, the row of columns will change to different colours listed in the array.

## Add Pause and Resume function

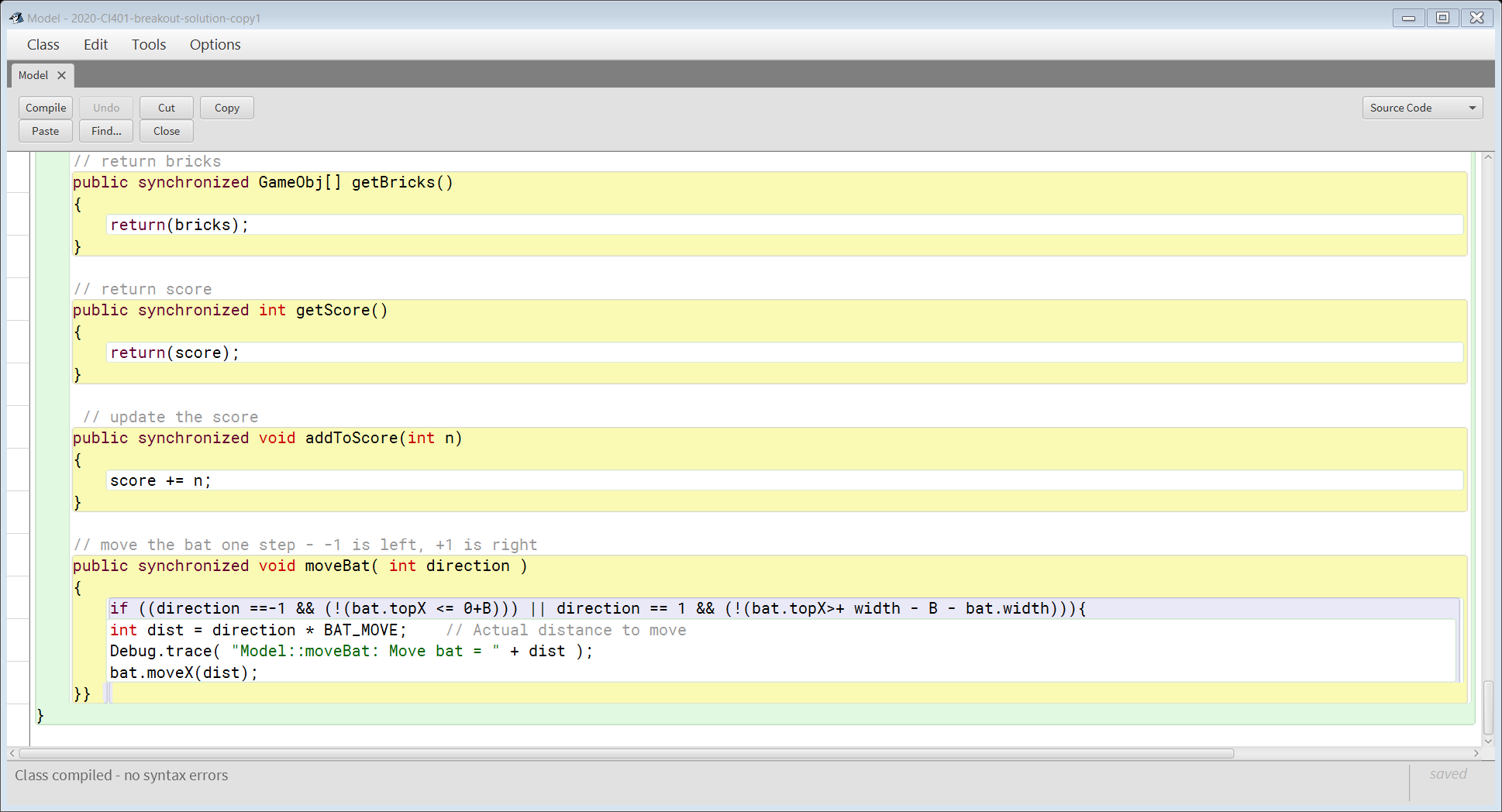


When creating the Pause and Continue Function, I wanted players to use a particular key for the functions to occur. Therefore, under Controller, I had made switch cases P (Pause) and C (Continue) to run the function. When thinking about how the game should pause, I had thought how other moving Game Object should freeze for the game to be paused. Therefore, when identifying a variable or function which would freeze the component, I had noticed that under the model class there was variable which stated the units for the ball movements. With this variable I was able to control the speed of the ball. With this idea, I had inherited the following functions below to the controller class. This will allow me to configure the variables of the speed.



Under the case P, I had configured the units of the ball and bat movement to 0 for the bat and ball current position to be still when the key is inputted. Under the case C, both variables are changed back to the original values of the game.

## Fixing the Bat



To prevent the bat moving out of the game, I had thought that the if statement would suit to achieve the aim. The bat movement was under the Model class. For the condition in the if statement, I wanted it to compare the direction of the bat also the position of the bat to the border of the edge of the panel as stated in the above screenshot. The two different condition would vary based on which end of the game the bat was positioned. This if statement would the move the direction of the bat -1 or +1 based on the position of the bat. In order to find the position of the bat, I included the extension of topX.