Breakout Project

CI401 Introduction to Programming

Cameron Stowell

Contents

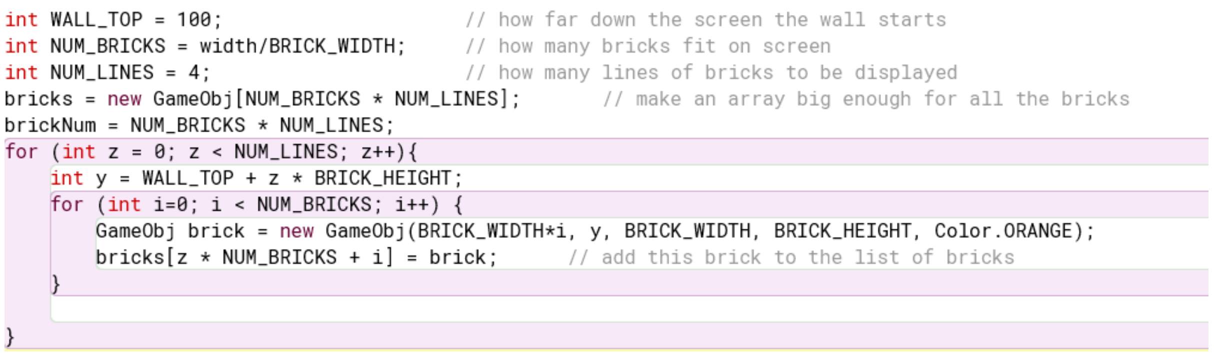
Introduction

My goal with this breakout game is to make it more professional in terms of quality. After a few goes at the game I noticed some glaring problems. The bat would continue moving even though it was not visible on screen. This makes players see it as unfinished. Once the game is completed (there are no bricks left on screen), nothing happens. This is a huge problem in two ways. Firstly, the player may not think the game is over and so there is no reward for completing the game. This takes away any incentive to play again. I have an idea which displays different lines of text depending on how high your score was, this gives an incentive to play again to beat your previous score. Secondly, the ball continues to bounce around despite any bricks remaining, this makes it impossible for the player to increase their score further, yet the ball can collide with the bottom of the screen to reduce the score. This is very unfair for the player and makes the game less fun to play. I would like to add a game over screen that displays the score, whilst preventing the ball from moving. I also noticed that when a ball hits a brick the score increases by 100 but when the player misses the ball it decreases by 200. It is an unfair for the player once again as it requires minimal mistakes to be made. I would rather the score be more favourable for the player to make a more relaxed and fun game. Looking through the code revealed to me a built-in fast mode where the ball moves faster. I think that would be a good feature to make it a little harder for some players who want a challenge. I have multiple ideas to improve this game and aim to increase its quality.

Design and Development:

Adding more bricks:

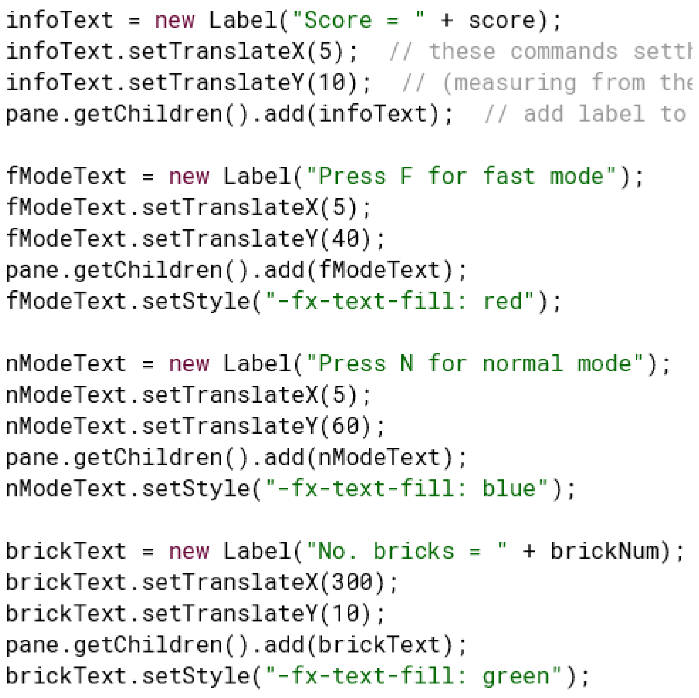
One row of bricks isn't enough for a breakout game, it isn't very fun to play as the game is over very quickly. Therefore, I decided to add a few more rows to at least make the game last longer. This was all done in one chunk of code. I started by making a new variable called NUM_LINES. This evidently is the number of rows of bricks that will be on the screen, I have it set to four, but it can be changed. Then an array is made by multiplying the number of bricks by the number of lines. The first for loop creates a new variable: z which is set to 0. Then another requirement is set, the loop only continues if z is lower than the number of lines which is four in this case. Finally, every time a loop is started, z gets 1 added to its value therefore the loop will run four times for the four lines of bricks. Inside the first for loop is y which is another variable meant to represent the y axis, this makes the bricks place underneath each other. Without this the bricks would place, but stack on top of each other four times. The next for loop is a similar set up as it loops until the number of bricks is reached for the row. It places a brick per loop and will loop 12 times. Once that is complete the first for loop repeats causing the row to drop a level and another twelve to be placed.



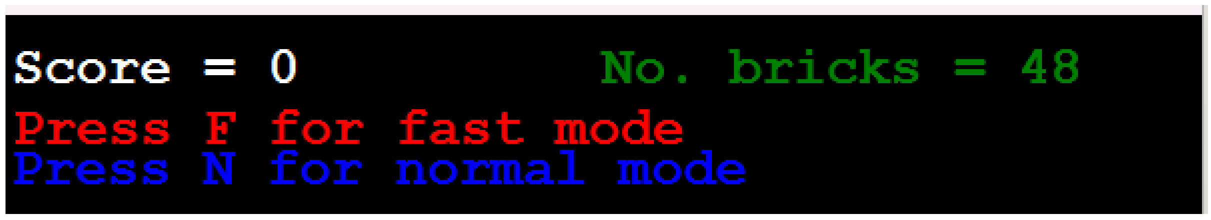
```
int WALL_TOP = 100;                    // how far down the screen the wall starts
int NUM_BRICKS = width/BRICK_WIDTH;    // how many bricks fit on screen
int NUM_LINES = 4;                     // how many lines of bricks to be displayed
bricks = new GameObj[NUM_BRICKS * NUM_LINES];       // make an array big enough for all the bricks
brickNum = NUM_BRICKS * NUM_LINES;
for (int z = 0; z < NUM_LINES; z++){
    int y = WALL_TOP + z * BRICK_HEIGHT;
    for (int i=0; i < NUM_BRICKS; i++) {
        GameObj brick = new GameObj(BRICK_WIDTH*i, y, BRICK_WIDTH, BRICK_HEIGHT, Color.ORANGE);
        bricks[z * NUM_BRICKS + i] = brick;      // add this brick to the list of bricks
    }

}
```

Text placement and Colour:

When exploring the code, I noticed that there was a fast mode for the ball. When set to true the ball moves much faster which makes the game harder. Therefore, I decided to add some text to let the player know they can press the F key to speed it up. To do this I had a look at how the score text was implemented. It involves using setTranslateX and setTranslateY to place the text at specific x and y coordinates in the window. I did some research on how to change the text colour using java fx and I found the answer on stack overflow [1]. "-fx-text-fill: blue" will set the text assigned to a variable to blue. Blue can be swapped to any colour for the same effect. I added other text elements such as how to revert the fast mode and a brick counter for progress.



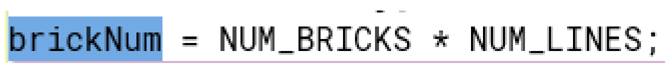
```
infoText = new Label("Score = " + score);
infoText.setTranslateX(5);  // these commands sett
infoText.setTranslateY(10);  // (measuring from the
pane.getChildren().add(infoText);  // add label to

fModeText = new Label("Press F for fast mode");
fModeText.setTranslateX(5);
fModeText.setTranslateY(40);
pane.getChildren().add(fModeText);
fModeText.setStyle("-fx-text-fill: red");

nModeText = new Label("Press N for normal mode");
nModeText.setTranslateX(5);
nModeText.setTranslateY(60);
pane.getChildren().add(nModeText);
nModeText.setStyle("-fx-text-fill: blue");

brickText = new Label("No. bricks = " + brickNum);
brickText.setTranslateX(300);
brickText.setTranslateY(10);
pane.getChildren().add(brickText);
brickText.setStyle("-fx-text-fill: green");
```



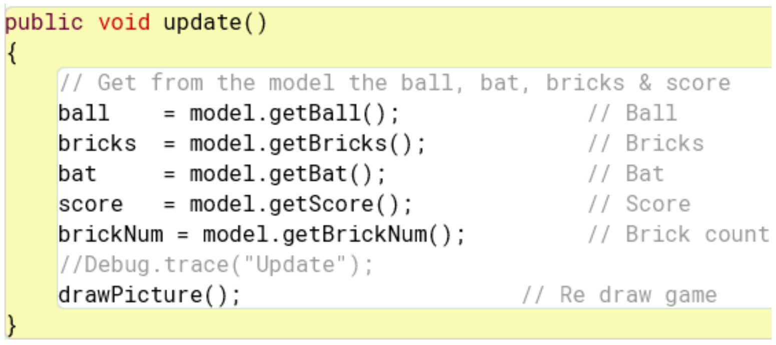
```
Score = 0          No. bricks = 48
Press F for fast mode
Press N for normal mode
```

Brick counter:

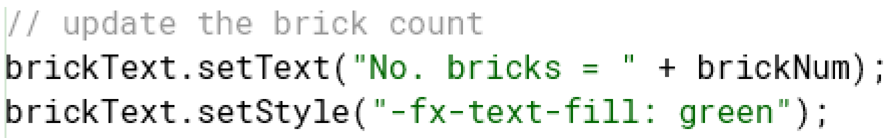
To add the brick counter code was required between classes. In the model brickNum was calculated. Once the brickNum was calculated it was then reassigned in the view class. Using model. getBrickNum it takes the value from the model and assigns it to the same variable but in the view class. This is seen below. I then added the text to the game view using the same method as the other text in the previous explanation. This is shown below. This adds text that says: "No . bricks = 48" in green text. The problem is the brickNum doesn't decrease when bricks are hit, it's just a flat number. To combat this, I had a look how the score updated when a brick was hit and replicated it to fit the brick number. Underneath the score addition I added the brick count variant. addToBrickCount is a method and hitBrickNum is a variable set to 1. Then using the synchronized method and -= the brick count will now reduce by one when a brick is hit and the value is synchronized with other classes so it updates live. Now the brick count reduces accurately when a brick is hit.



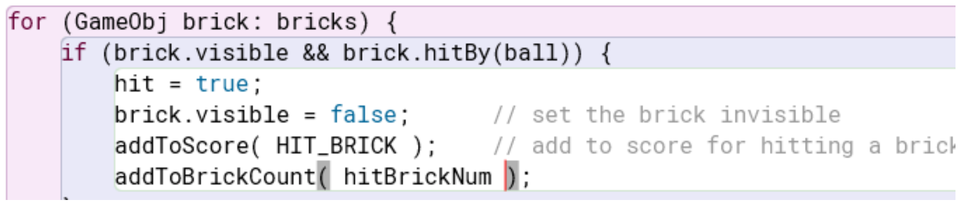
```
brickNum = NUM_BRICKS * NUM_LINES;
```



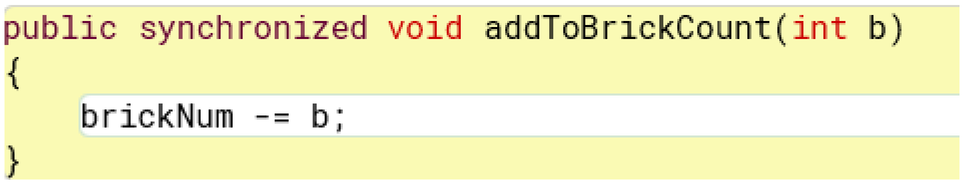
```
public void update()
{
    // Get from the model the ball, bat, bricks & score
    ball     = model.getBall();            // Ball
    bricks   = model.getBricks();          // Bricks
    bat      = model.getBat();             // Bat
    score    = model.getScore();           // Score
    brickNum = model.getBrickNum();        // Brick count
    //Debug.trace("Update");
    drawPicture();                     // Re draw game
}
```



```
// update the brick count
brickText.setText("No. bricks = " + brickNum);
brickText.setStyle("-fx-text-fill: green");
```



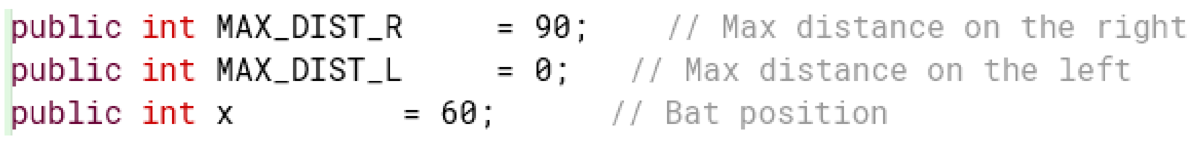
```
for (GameObj brick: bricks) {
    if (brick.visible && brick.hitBy(ball)) {
        hit = true;
        brick.visible = false;      // set the brick invisible
        addToScore( HIT_BRICK );    // add to score for hitting a brick
        addToBrickCount( hitBrickNum );
```



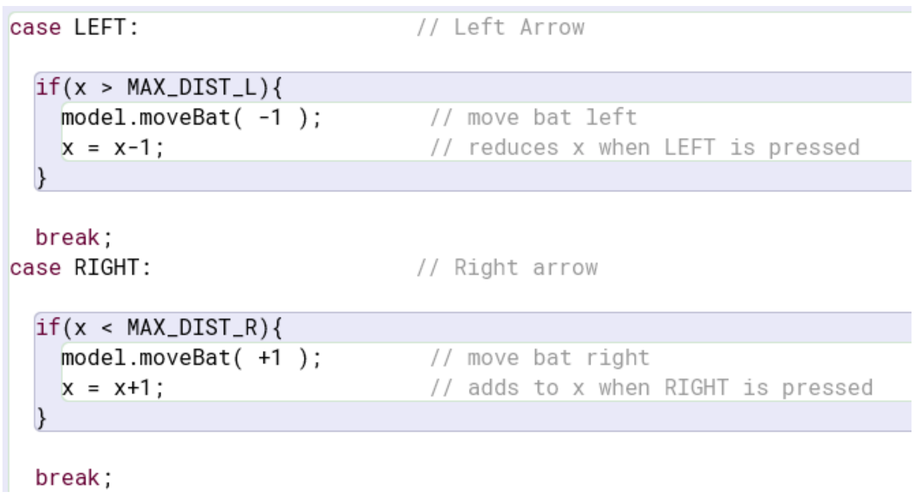
```
public synchronized void addToBrickCount(int b)
{
    brickNum -= b;
}
```

Preventing off-screen bat movement:

The bat seems unfinished due to it being able to pass the boundary of the window and continue to disappear off-screen. To limit the bats movement, I had the idea to add a coordinate system. This means setting a maximum distance on the right side and a maximum distance on the left side. The variable x is the position of the bat when the program is run. Once the variables were set, I needed to find a way to change them when an input is made. An if statement is used to check that the bat isn't past the maximum distance on either side. Once it confirms it isn't passed the max distance it allows movement, and every input adds 1 to x if it goes right and removes 1 when the left key is pressed. This tracks the location of the bat and prevents left movement when too far on the left and the same for the right side. The values for MAX_DIST_L, MAX_DIST_R and x were found using trial and error due to the bat starting slightly to the right of the centre.



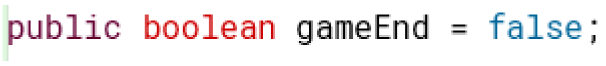
```
public int MAX_DIST_R      = 90;     // Max distance on the right
public int MAX_DIST_L      = 0;    // Max distance on the left
public int x           = 60;      // Bat position
```



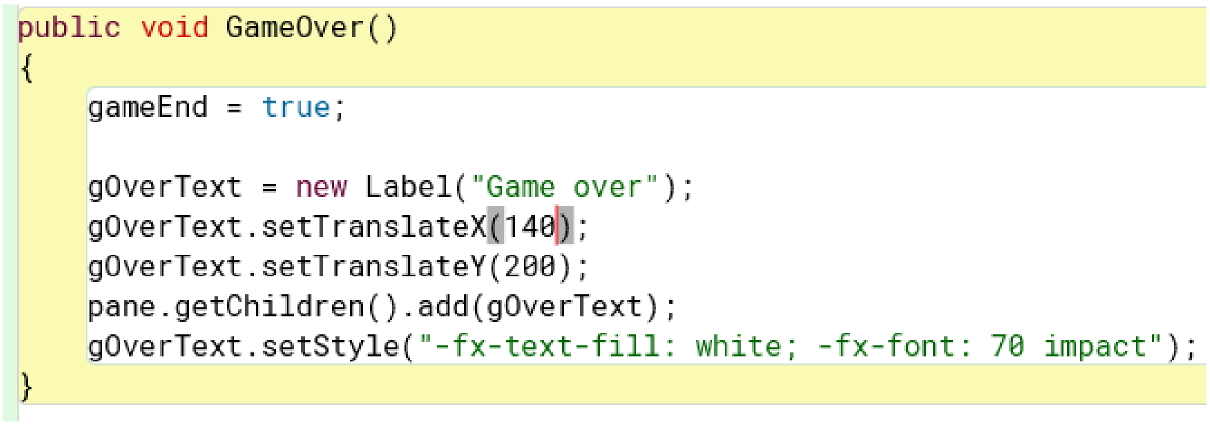
```
case LEFT:                          // Left Arrow

  if(x > MAX_DIST_L){
    model.moveBat( -1 );            // move bat left
    x = x-1;                        // reduces x when LEFT is pressed
  }

  break;
case RIGHT:                         // Right arrow

  if(x < MAX_DIST_R){
    model.moveBat( +1 );            // move bat right
    x = x+1;                        // adds to x when RIGHT is pressed
  }

  break;
```

Game over when there are zero bricks:

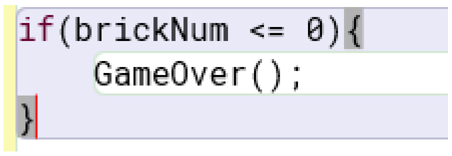
To let the player, know the game is over when the brick counter reaches zero, large white game over text appears in the middle of the screen and the ball and the bat disappear. Firstly, a Boolean value needs to be made. This is gameEnd and it is set to false by default. Then I created a method called GameOver which sets gameEnd to true and prints the label “Game Over” in the middle of the screen. I don’t want the GameOver method running earlier than expected so I created an if statement in the DrawPicture method, so the brick count must be 0 for the GameOver method to run. Once it has run gameEnd gets set to true instead of false this then runs an alternative part to the DrawPicture method. Therefore, the bat and ball stop displaying on the screen and the score stops updating so the final score is decided. I did some research on how to hide text as all of my text remained after the game has finished. setVisible sets the text to visible so when false is implemented the text is hidden. This makes the game over screen only the game over text. I then added another label which tells the player their final score underneath the words game over. Model.setGameState sets the game state to finished which prevents the ball from moving and so the score can’t change. Finally, I added special dialogue which vary depending on the score. 2000 or below is classified as a low score and has bronze text colour. 2001 to 4000 is a good score and has silver text and 4001 and above is an amazing score with gold text.



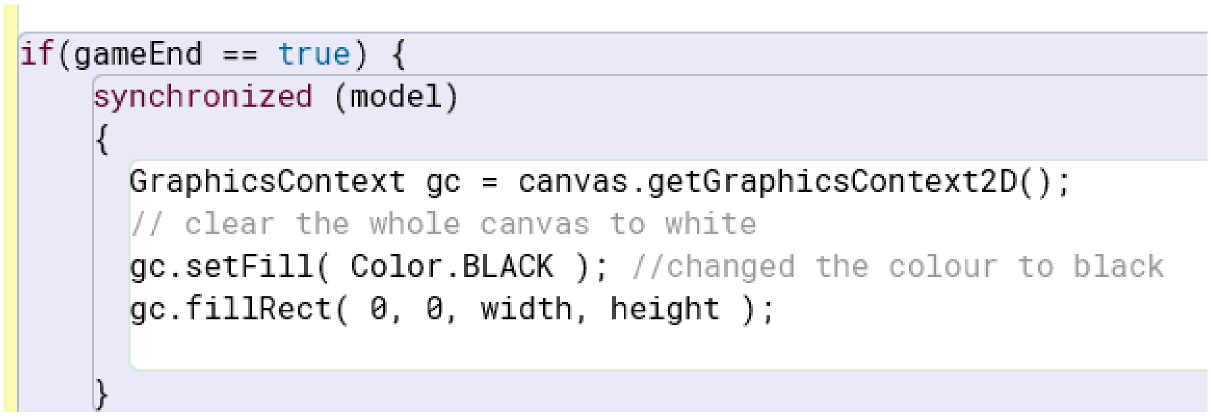
```
public boolean gameEnd = false;
```



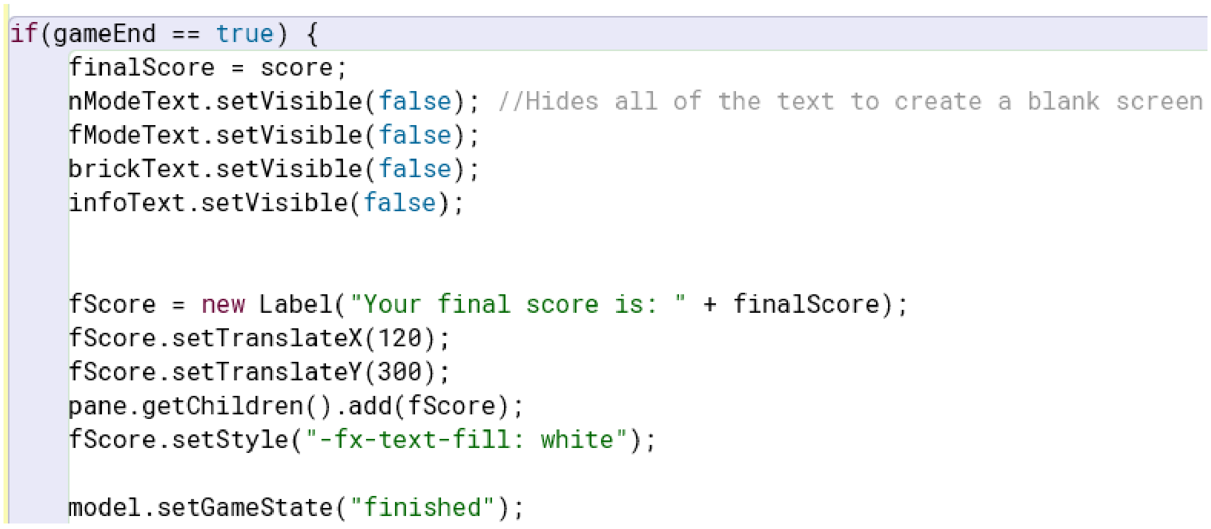
```
public void GameOver()
{
     gameEnd = true;

     gOverText = new Label("Game over");
     gOverText.setTranslateX(140);
     gOverText.setTranslateY(200);
     pane.getChildren().add(gOverText);
     gOverText.setStyle("-fx-text-fill: white; -fx-font: 70 impact");
}
```



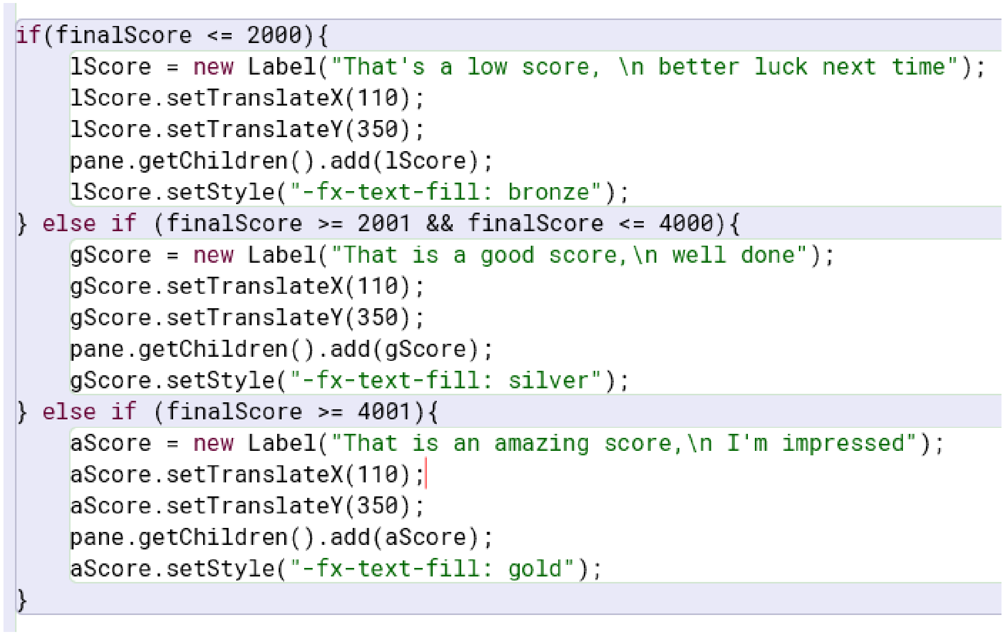
```
if(brickNum <= 0){
    GameOver();
}
```



```
if(gameEnd == true) {
    synchronized (model)
    {
        GraphicsContext gc = canvas.getGraphicsContext2D();
        // clear the whole canvas to white
        gc.setFill( Color.BLACK ); //changed the colour to black
        gc.fillRect( 0, 0, width, height );

    }
```



```
if(gameEnd == true) {
    finalScore = score;
    nModeText.setVisible(false); //Hides all of the text to create a blank screen
    fModeText.setVisible(false);
    brickText.setVisible(false);
    infoText.setVisible(false);


    fScore = new Label("Your final score is: " + finalScore);
    fScore.setTranslateX(120);
    fScore.setTranslateY(300);
    pane.getChildren().add(fScore);
    fScore.setStyle("-fx-text-fill: white");

    model.setGameState("finished");
```



```
if(finalScore <= 2000){
    lScore = new Label("That's a low score, \n better luck next time");
    lScore.setTranslateX(110);
    lScore.setTranslateY(350);
    pane.getChildren().add(lScore);
    lScore.setStyle("-fx-text-fill: bronze");
} else if (finalScore >= 2001 && finalScore <= 4000){
    gScore = new Label("That is a good score,\n well done");
    gScore.setTranslateX(110);
    gScore.setTranslateY(350);
    pane.getChildren().add(gScore);
    gScore.setStyle("-fx-text-fill: silver");
} else if (finalScore >= 4001){
    aScore = new Label("That is an amazing score,\n I'm impressed");
    aScore.setTranslateX(110);
    aScore.setTranslateY(350);
    pane.getChildren().add(aScore);
    aScore.setStyle("-fx-text-fill: gold");
}
```

Critical review

Three reasons why my code additions are good:

1. **Game completion**. Adding a game over screen was a necessary addition. Without it the game isn't as rewarding. There is no sense of completion and no reason to play again. Telling the player their score and how good it is could encourage them to keep playing to improve.
2. **Bat movement restrictions**. Preventing the bat from leaving the boundaries of the screen is also very important. The game feels very unfinished and unprofessional when it disappears off screen. The game feels more professional and polished since I made this change.
3. **Brick layers**. The addition of more layers of bricks is another important change. The game was way too simple and easy with only twelve bricks to hit. It would end very quickly with little to no effort from the player.

Three points where the code could be improved:

1. **Brick layout and look**. I think the bricks could have an outline to them, so they are easy to identify individually. I also do think different brick layouts with gaps and special bricks could make the game more interesting. The individual bricks could also have different colours so it's more visually appealing.
2. **High score leader board**. Alongside the score system which has different dialogue depending on performance, a scoreboard of the top 5 scores that you have gotten could encourage more playing to get the highest score possible.
3. **Pause menu.** I attempted to create a pause menu with a similar idea of how to end the game. Setting the game to finished would prevent ball movement so the score doesn't change however, there was no way to revert the pausing effect where the ball continues its path. This would have been a nice improvement so the player can pause to take a break.

Conclusion:

Throughout the development of my game, I have learned multiple important parts of Java. When preventing the bat from moving off the screen I had spent multiple hours with the right concept, but it wasn't working. I then decided to move a variable I had assigned in the method to the class, and it suddenly worked. It showed me that even where the variable is assigned can make a huge difference with the function of the program. Similarly, when developing the game over feature, I had tried to put "model.setGameState("finished");" in the main class but it didn't work. However, once I put it in the view class within the drawPicture method it worked. Whilst adding the brick counter I had to declare the variables as public, as both the view class and the model class were required to add the brick counter. Therefore, I learnt how to code using multiple classes and access modifiers. I also learnt how to position text using the x and y axis with "setTranslate" this was very helpful as I added multiple lines of text both while the game was running and once it has ended. After researching I figured out how to hide text using ".setVisible(false)". This was found on Stack Overflow [2]. A feature I discovered when playing around with bluej was the find button. I used this a lot throughout to help me locate where variables are so I could understand how the code works. Overall, I have learnt a lot about java and javafx from experience with this project. I have improved my coding skills dramatically.

My self-assessed grade:

Development: A Report: C

References:

1. Pfeffer, M. (2015) *How to change the colour of text in javafx Textfield?, Stack Overflow*. Available at: https://stackoverflow.com/questions/24702542/how-to-change-the-color-of-text-in-javafx-textfield#:~:text=If%20you%20are%20designing%20your,text%20color%20of%20your%20Textfield%20 (Accessed: 14 May 2023).

2. Uchiha, I. (2015) *How to show and then hide a label in javafx after a task is completed, Stack Overflow*. Available at: https://stackoverflow.com/questions/31607656/how-to-show-and-then-hide-a-label-in-javafx-after-a-task-is-completed (Accessed: 14 May 2023).