BREAKOUT

Zac Ioannidis

REPORT

Author

Zac Ioannidis || zi12467

Analysis

This program is a reimplementation of the classic Breakout game, which was released in 1976 by Atari. The player bounces a ball on a paddle controlled by the keyboard, which bounces off the top and side walls of the screen. Rows of bricks exist at the top of the screen which are destroyed when the ball hits them. Points are awarded to the player for each brick destroyed and when all are destroyed, the next level, which is randomly generated, is presented. The player has 3 lives, which are lost when the ball touches the bottom of the screen.

The player is presented with a main menu at the start of the game, which starts when <SPACE> is pressed. If the player loses, he has the option to restart. As in the original, there is no end to the game. In this implementation, levels are randomly generated, meaning that the game ends only when the player dies.

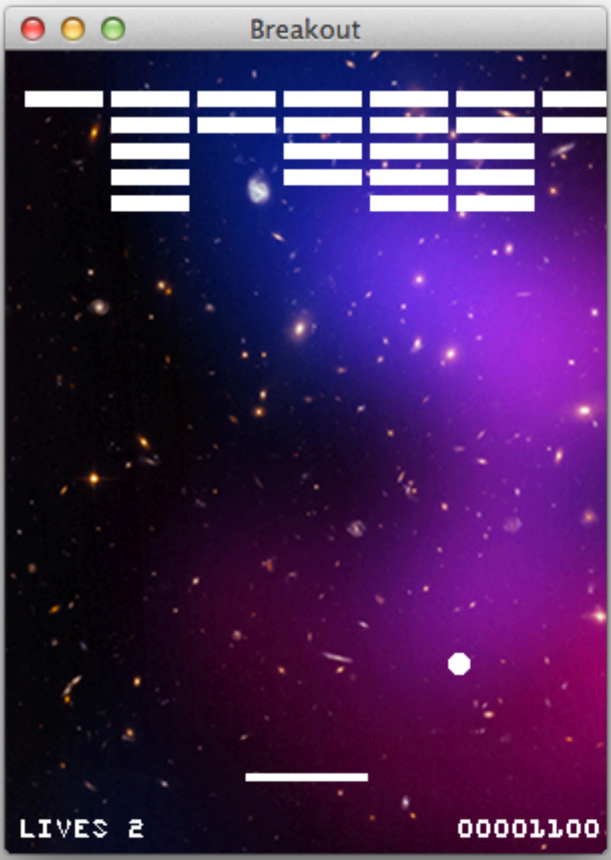
Design

The game consists of several components. Game objects include the paddle, ball and bricks, which are implemented as separate classes and include all of their respective properties, namely their coordinates and various others metrics, such as their width and their height. Other properties of the classes include horizontal and vertical speed for the ball, movement speed of the paddle and state of each brick (whether it is destroyed).

The entry point of the game is the Play class, which inherits and extends a JFrame. It specifies the properties of the window (title, dimensions, initial location relative to screen) and sets the content pane to a fixed height and width, since the borders of the frame are dependent on the operating system running the program.

The Board class, a JPanel is then added to the frame. This class contains all the game logic needed to play the game and handles the drawing of the game elements to the screen. Bricks are initialized and stored as an *ArrayList* and when a brick is destroyed it is removed from the data structure. Therefore, the paint() method no longer draws it on screen. When the *ArrayList* is empty, meaning all bricks have been destroyed, the game progresses to the next level, which features a randomly generated brick formation.

The ball, which is an Ellipse2D object is accurately described by its horizontal and vertical speed. When a brick or the paddle is hit, the vertical orientation of the ball changes and, in a similar fashion, the ball hitting the side of the screen results in a change in its horizontal orientation. To allow better control of the ball and enable better gameplay, the direction and horizontal speed of the ball after it is hit by the paddle is specified by the part of the paddle that intersects it.



There are three distinct game states, which are handled by boolean variables, the main menu, game over and game paused states. Transition from the main menu to the game is done by pressing <SPACE> and <ESC> is used to pause the game. When the player has lost, he is presented with the choice to restart the game by pressing <SPACE>.

Rules are introduced to specify the position of the ball after it has hit a wall, a brick, or the paddle and are included in the Board class, alongside the game logic. Its vertical orientation is inverted after hitting the latter two components. Paddle and ball movement are defined in their respective classes.

There is also a class which includes inter-class variables, such as the height and width of the frame's content pane and whether the game is paused or not. This class was first implemented as an interface, but the need to change the variables as the game progressed and its lack of adherence to Object-Oriented design patterns introduced the need for it to be refactored and written as a class.

Implementation

The game is written in Java and is a standalone application containing six classes, with one of them being the entry point. The files are not packaged, as they will be compiled and run from the command-line.

Testing

Extensive testing was done by playing the game and covering all possible possibilities that could be encountered during the course of a game session. The game was also given to beta testers to play and their feedback was taken into consideration during the development of the game. Furthermore, unit tests were written for the classes containing game logic and those that contain data and do calculations. *System.out.println()* calls were used with extra consideration regarding their interpretation because of the game's use of threads.

Development

This program is a standalone Breakout clone, the world-famous game first developed by Atari and prototyped by Steve Wozniak. Several important features were left out of the game during its development due to time constraints and will be introduced in the future.

There are two bugs currently in the game. When the player destroys 2 bricks at the same time, a third one, seemingly random, is destroyed as well. This happens because they are removed from the ArrayList, which is referenced by another thread milliseconds afterwards. But, of course for the time being this is a feature and not a bug. The most urgent bug-fix includes packaging the fonts used in the game with the source files to enable a uniform experience to anyone playing the game from a variety of different operating systems. In this installment of the game, the two fonts used in the game are provided along with the source code.

Potential enhancements to the game include the ability for new levels to be created by reading in a text file and the introduction of various power-ups, such as extra lives, slowing down the ball or faster paddle movement. There could also be an added audio component to the game, such as a soundtrack and sound effects for when the paddle hits the ball.