Project: Brick Game

Group member: Minxiang Pan(mp3335), Zeyu Dong(zd2162)

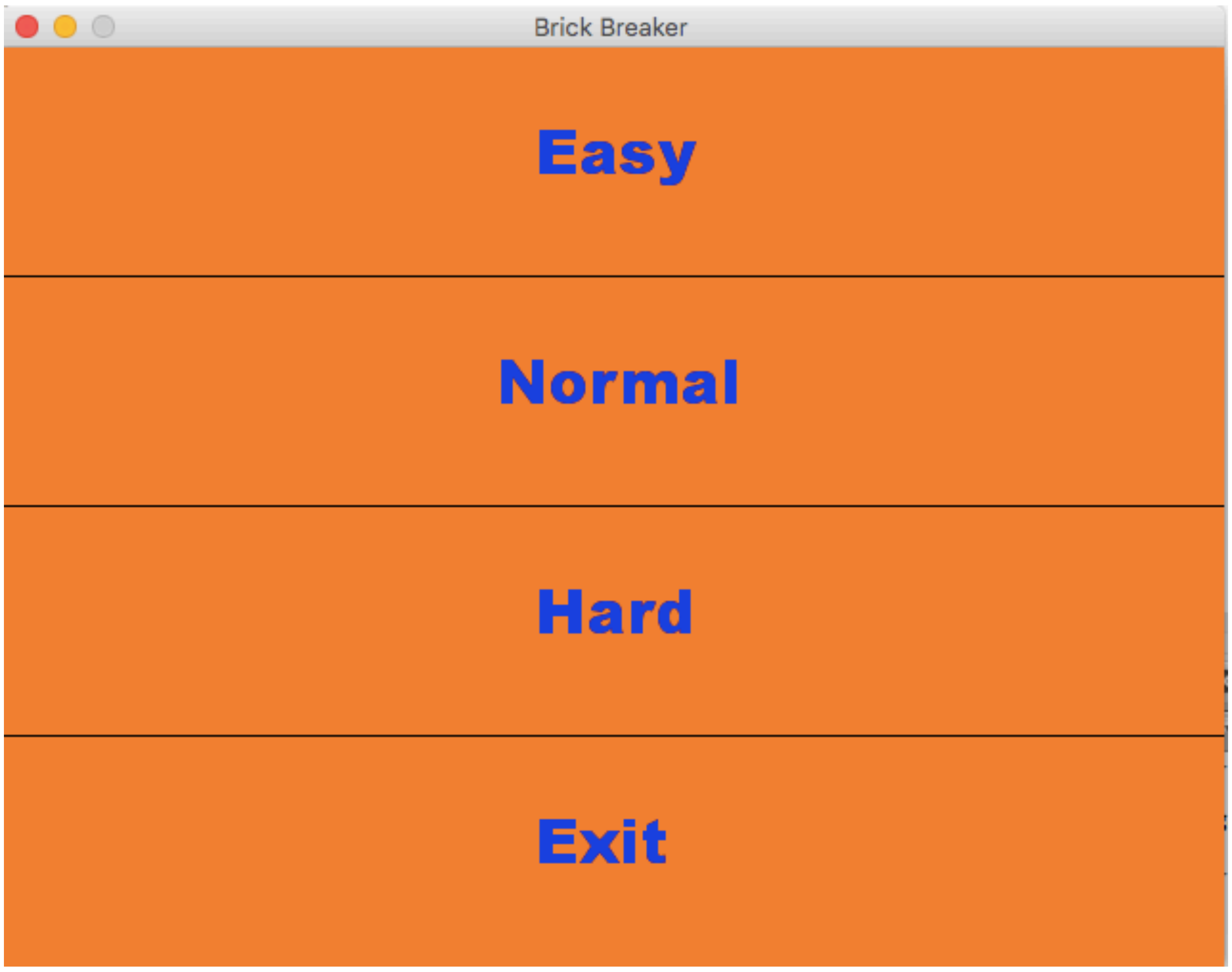
NOTICE: Change the path of background image and music and use python 2.7 before trying.

What does it do?

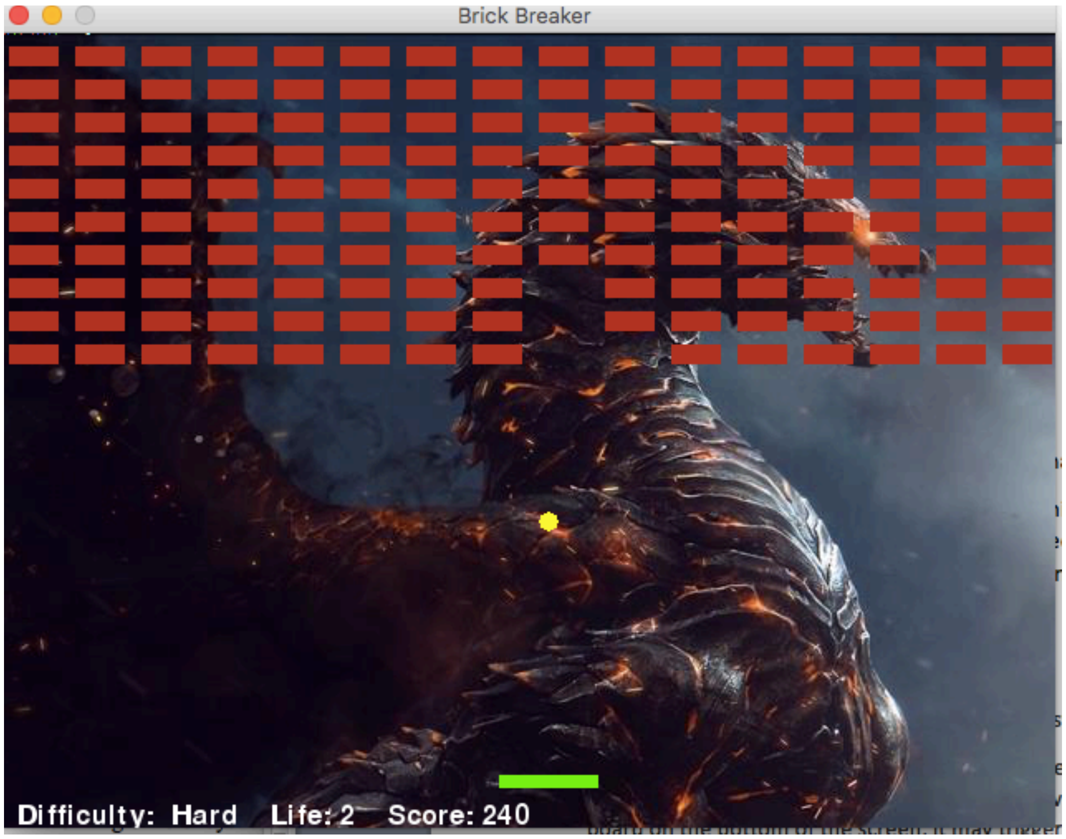
This is actually a brick game. When the game begins, you can shoot the ball first. And then, the ball may go up in a certain slant line. Once the ball hits the board above, the board the ball hits may disappear and the ball may bounce back in a certain slant line. When the ball reach the bottom of the screen, you must make the ball hit the board which is on the bottom of the screen, so that the ball can bounce back and "play" another round.

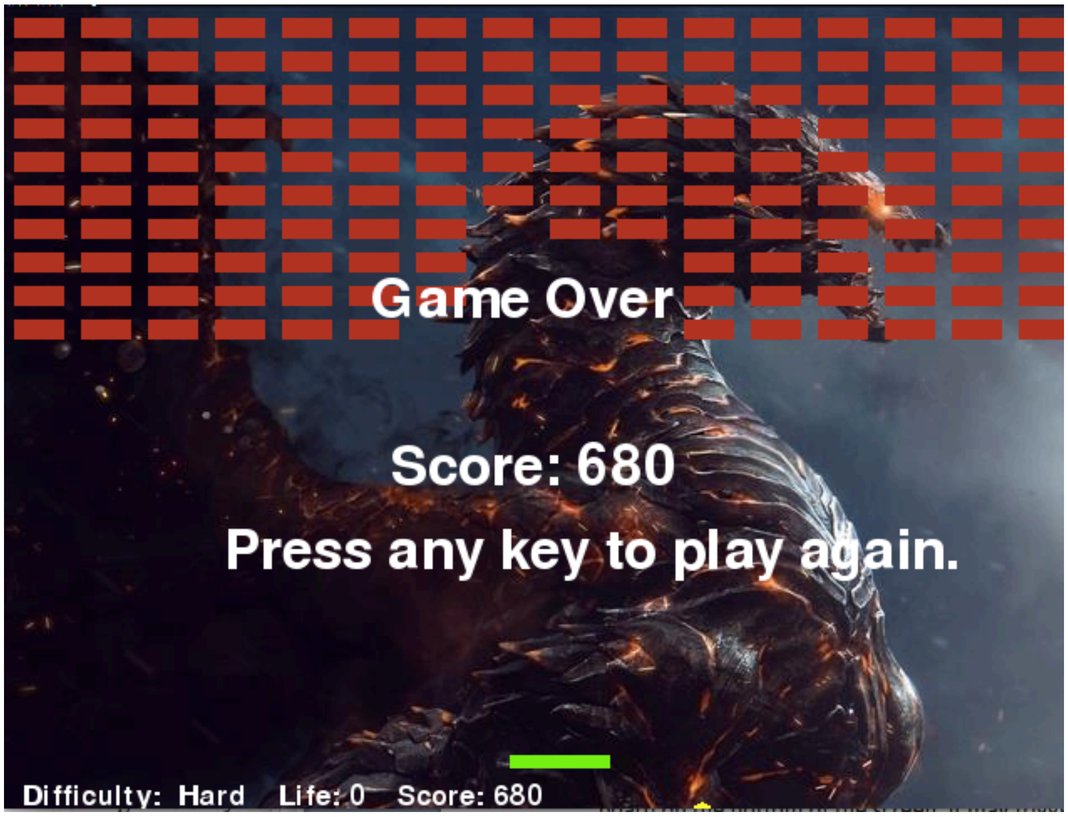
The main goal is "eliminating" all the boards on top of the screen. During the game, if you fail to make the ball hit the bottom board, it means you lost one chance. And there are certain chances for you to play the game. Once you lose all the chances, it means you lost completely, and you will be brought to the main menu where you can choose the difficulty of this game.

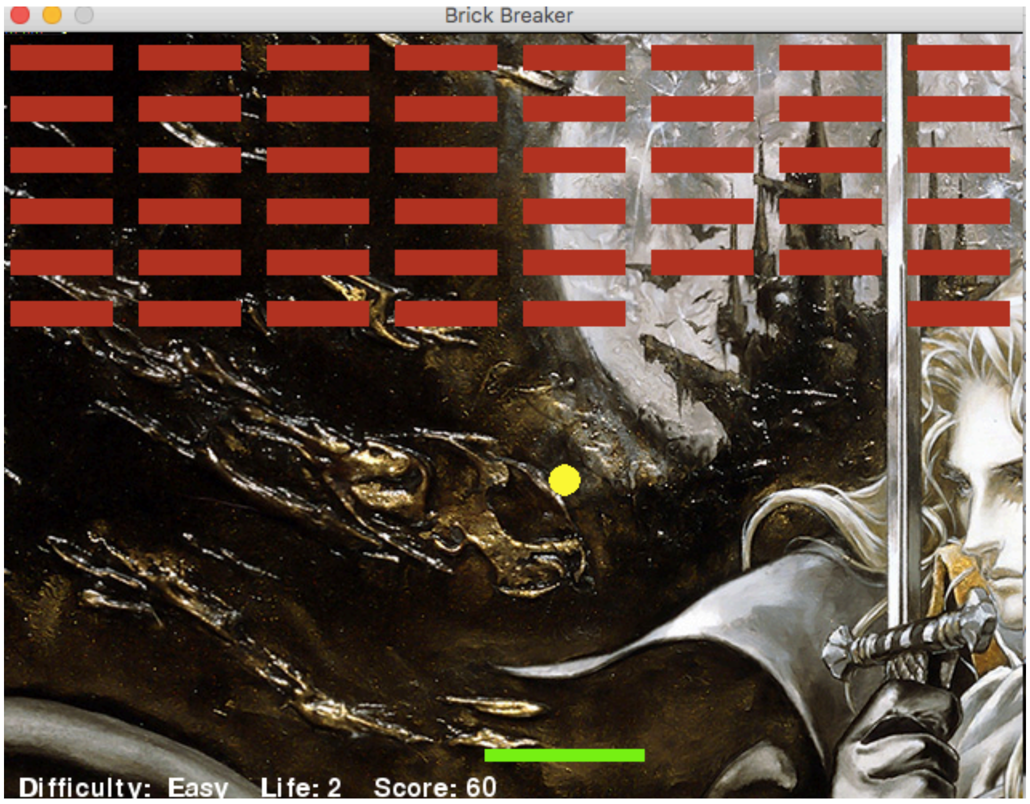
Game Menu:



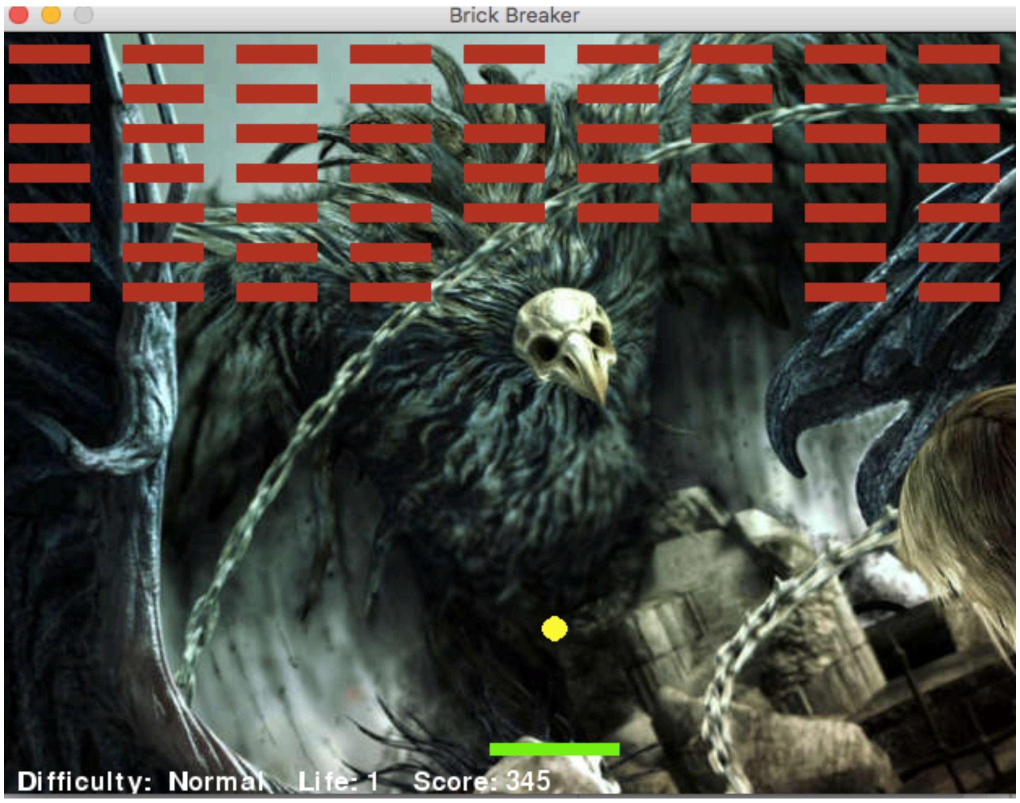
Game picture:




Game Over
Score: 680
Press any key to play again.
Difficulty: Hard   Life: 0   Score: 680



Difficulty: Easy Life: 2 Score: 60



Difficulty: Normal Life: 1 Score: 345

Architecture of your project - how is the functionality divided up? What classes are defined?

The project is mainly divided in three parts: definition of key variables, definition of key functions and main loop. No classes are necessary to this project, because we use two 'while loops' plus well defined functions to realize the main loop of the brick game.

How to run your program? Which version of python are you using?

We import pygame under Python 2.7 to build up our project, because the pygame for Python 3.4 we've found does not work very well for our project.

Command line arguments? What extra modules does it require?

No command line arguments is needed, all the configuration work would be initialed in the main loop of our program. We also need 'time' module to set a wall-clock in the main loop to process time. And 'random' module is also necessary for setting the variation of the movement of the ball after every hit to either the bricks or paddle. In addition, 'sys' module is used as 'sys.exit()' to quit the game.

Discuss your experience with the project:

We had an interesting experience with our project using Python. Generally speaking, it is the first time to develop a game by ourselves. The 'Pygame module' has provided an excellent platform for the rookies to design their own game. In the module we could use Pygame.init() to initial the environment of the game; create our window surface using pygame.display.set_mode() where everything is drawn on this surface; add event listener by 'pygame.event' to make an interaction between the user and game through keyboard. 'Pygame.draw' is utilized for drawing our bricks, paddle, ball and 'screen.blit' is for inserting our beautiful background image; also 'pygame.mixer.music' is used for adding the exciting background music; last but not the least, we use 'Pygame.display.upgrate' as a tool for 'refresh' the window in the while loop to realize the movement of the game.

Several roadblocks have been met through our project, but fortunately most of them have been solved with our effort. The first thing is the management of our variable. Since there are more than thirty variables for this game, an efficient way of managing the name of variables is important. So we use the format of defining new variable like 'OBJECT_ATTRIBUTE' such as 'PADDLE_WIDTH', 'BALL_X'. For another thing is the design of brinks. We have experience a hard time until we finally come up with an idea to use an array to store whether the brick is hit or not. Still another thing is the change the movement of the ball, which is the most difficult part in our project. We cited part of the idea in other game on http://blog.csdn.net/guzhou_diaoke/article/details/8244803. In order to realize this part, first we have to check the collision for the ball, paddles and bricks. The algorithm is clear that we make judgment for the collision if and only when the ball plus its size fall within the width for either the paddle or bricks. Also, it would bounce back when the ball hit the side of the screen. Next, we have to upgrade the status of the brick array once the brick is hit. Also, if the ball falls out of the boundary of the paddle, we have to subtract one life from the total. The last difficult things are the design for the menu and configuration of the game. We used ‘GAME_INIT’, ‘GAME_RUN’, ’GAME_OVER’ to keep track of the game status. Once the game is over, it would return to the menu to select the difficult level, which has been mentioned in previous part. The design of the menu is to make it more convenient for users to choose the mode they like, and under different difficulty the configuration for parameters such as game speed, number of bricks, length of paddle etc.

However, one thing has not been solved yet. Every time when I exist the game and want to try again, I have to re-launch the Spyder otherwise some errors would occur saying ‘kernel died unexpectedly’. So this is a limit of our program.

What extensions to your project might be interesting?

One extension that can make the game more interesting is that when the ball hits the boards on top of the screen, there may be some “items” goes down from the board the ball hits. If the “items” hit the board on the bottom of the screen, it may trigger some special random events. For example, the ball may become faster (we can change the speed of the ball), or the ball may become smaller or the width of the board may become narrower or the number of balls may increase to three. Those special events can make the game more enjoyable.