Final Report

CS443 Group Project

Brick Smasher

Submitted by,

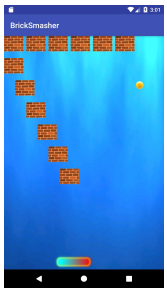
Kushal Khanal

Kashyap Soni

12/19/2017

Project Statement

We wanted to build a gaming application, where all the individuals can play it, and use it for entertainment purpose. The name of our application is Brick Smasher. It is an arcade game, developed and published by Atari, which was conceptualized by Nolan Bushnell and Steve Bristow, and built by Steve Wozniak, and by Steve Jobs. And this game was first released in May 13, 1976. The application looks like the picture below, where it has four major parts. The ball, paddle, bricks, and background. The application does not have any special requirement.

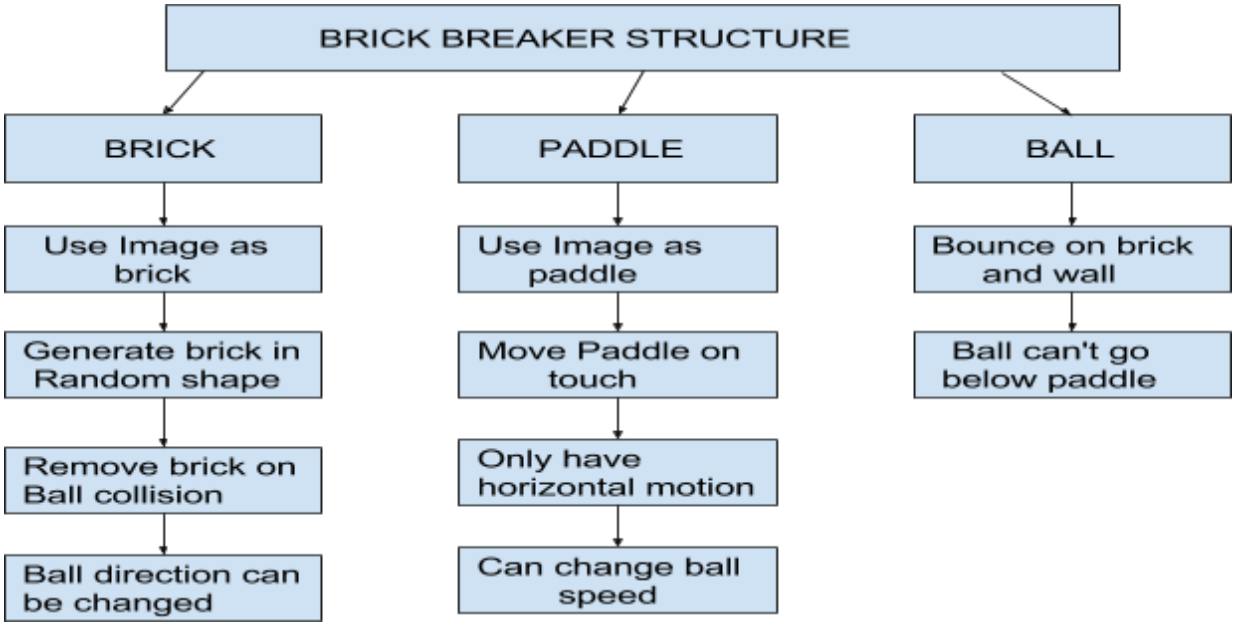


This picture shows how the ball, paddle, and brick will work upon the background.

Our targeted audience was basically kids but also adults and old people can play the game. This game is very challenging and it can create a deep affectionate quickly. We choose this because it was close to our childhood memories, and it was the best game back then to play.

Application Design

This application can be used in both the tablet and a smartphone. No outside service is used in this, but in future this might have a future to share their score to different web services. This application has 3 lives and a meter for score. Each time the player smashes a brick the score adds 20 points to the score and if the ball goes below the paddle then the player loses a live. And after losing three lives the game will be over and the final score will display. And the player can restart the game again. The ball moves and changes the direction when it hits the ball, brick or the walls which are left wall, right wall, and the top wall. The ball consisted of some features, as the ball is a image and it is used as Bitmap image which is obtained by using bitmapFactory and drawn in canvas. And the ball physics is based on the Bouncing Ball example given in class. The ball moves and interacts with the top, and side walls, and with paddle, bricks. The following structure is how the entire project will be implemented step by step.



In the App there are two activity, one is menuActivity, and another one is mainActivity.

MenuActivity: This activity deals with the main menu part, it has a button “play” which launches the mainActivity utilizing the Intent functionality, and textview High Score, that keeps track of the current HIgh Score in the game. The layout of this Activity is taken care by content_menu.xml, activity_menu.xml, test.xml

MainActivity: Mainactivity is where all the game’s activity truly begins. The Main Activity utilizes BrickSmasherview.xml for layout implementation. Paddle is the only element of the game that is independently governed by main activity. When the player loses all three lives, the game return backs to the menu Activity.

Application Implementation and Evaluation

These are the main components of the game.

This is the image which was used in the application. The ball is an simple object, which has properties like, the four coordinates, x speed, y speed. Inside it, it also has move method, that updates, the current position of ball. In the process of moving, ball checks,if it has encountered any wall, or the bricks, and if it encounters brick, it makes the brick invisible, and change direction , if it encounters wall, it reverses the respective direction.

And the ball starts at the center of the screen, after the screen is touched once ball initially moves upward.


This is the image of the paddle. The paddle, is also an image and used as imageview which responds on touch. The paddle moves when it is touched and hold in the direction which is left or right, also the paddle changes the direction of the ball when intersects with each other. And the paddle also has the four coordinates, and update method, which updates is x-axis position based on the touch response of user. And the paddle needs to be touched to move, when it is touched, it invokes ontouchlistener, and then updates the current paddle position.


This is the image of the brick. And the brick is an bitmap similar to the ball and imported using bitmapfactory. The brick is similar to paddle changes the direction of ball, and the brick will disappears on collision with ball. And the score increases when the ball collides with the brick. Brick has the fours co-ordinates, and it has a method that updates if the brick is currently visible or not. The background is the underlying layer on top of which ball, brick, and paddle resides, and interacts with each other.

Implementation

When the game first starts The menuactivity is launched and it displays a button play, which is responsible for launching the main activity and starting the game. There is also a text view that displays the current high score, which at the launch of the game is set to 0. After the users presses the play button. Main Activity is launched, where it calls the BricksmasherView to

update the layout. The paddle is drawn as an imageview Object and placed on the bottom and center of the screen. The properties of paddle like its width, position is recorded inside the paddle Object which is based upon the Paddle class. Inside Bricksamsherview, we first create a ball object, bricks array to hold all the bricks, canvas object to draw canvas, and holder to ensure the drawing on on canvas carries out smoothly. There are two threads **backthread** and **brickthread**, and each of them have their own responsibility. The backthread primary responsibility is to draw the canvas and all the bitmaps. It draws ball, the background, and the bricks on the canvas, it is also responsible for tracking the position of the ball.

The **brickthread** is responsible for keeping track of all the bricks on the screen, and checking the interaction between bricks and ball. When the user touches the screen the ontouchEvent method is called, which sets the ball position, and calls the ball.move() method, in which the current position of the ball, and screen width and height is send. Inside the move method, we check for if the ball is in collision with either wall or the paddle, and changes the direction and speed based on the point of contact on the paddle. The move method returns true, if the ball is still above the paddle, whereas, returns false, if the ball goes below the paddle. The ballpos variable checks if the ball is alive or dead, if its dead, it then checks the live, and if all the lives are lost, the game goes back to the menuactivity. If the ball is still alive, in that case, the draw method is invoked, which draws all the bricks that are still visible, and the ball on the current position of the ball. When the ball comes in the contact of the brick, inside the brick thread, it is checked if the ball is in contact with the brick, if the brick is visible, then we change the direction of the ball, and make the brick invisible, and thus we update the score. The update

method for score and lives resides within main activity , and this runs on main UI thread, as other threads can not update the text views.

To test and debug this application, we utilized the debugger present in android studio, as well as running it in the emulator, as well as our android phone ("HTC 10").

References

As the game itself is based on the Atari Breakout, the main reference for its design and implementation is based on original game itself.

The main reference we used for the motion of the ball was from professor's, bouncing ball example. Furthermore, we received lot of help from Stackoverflow, to solve some of our issues we could not complete by itself. Like for example the motion of ball in various devices, as i some devices it was slower, while on some it was faster. We were able to solve this, with the help we obtained from stack overflow, which suggested us to consider the screen pixel density as a factor to make the speed device independent. Like this, there were many questions and issue, we were able to solve through stack overflow, and through help of our fellow classmate, Mr Nirmal Nepal.

Experience and Thoughts

Overall, it was great experience building this game, and we had lots of fun, and we were happy to implement multiple topics that we had learned throughout the semester.. We thought it was

easy to build games, but it turns out, the beauty of game is dependent on various minor details, which makes the game look smooth, and beautiful. We had issues mainly with implementing the physics of the game properly.

The three, ball, paddle and the brick are connected as they interact with each other when the ball hits with the paddle and brick, but we had some difficulties on the positions of the ball. Still there are many glitches regarding the collision and sometime get kind of annoying. Issues like which side of the brick is the ball colliding and which direction it should reverse in. And same with the paddle, which direction it hits and which direction it should leave with a specific speed. Sometimes the ball slows down and sometimes it goes faster, and when the ball interacts with the bricks it will just reverse it's direction and if it is in a faster speed it will collide with multiple bricks and smashes it. There needs to be improvement in collisions of bricks and paddle, other than that the application runs smoothly and, score and lives works perfectly fine with the game. Threads works systematically and the physics for the ball, brick, and paddle works accordingly fine. The ball changes its direction when it hits the sides walls and the top, and with brick and paddle, and if it goes below paddle you lose a live. And the ball start at the middle of the screen, which was a new thought for us, because usually the game starts with the ball attached to the paddle and when the paddle is touched it releases the ball. This ball will start at the middle and will always start moving from the right hand side. We wanted to implement more stages with different background and design, and with more speed of the ball to make the game more harder to play and interesting, but time limit was a issue and the interaction of ball with the brick and paddle was challenging for us to overcome. And still the ball does not work correctly when interacted with the brick and paddle.

We would like to thank professor Bo Sheng again for the amazing semester, we are really grateful for all the things he taught us, and by help of which we were able to build this app from scratch.