



Faculty of Engineering and Technology
Electrical and Computer Engineering Department

Software Development for Android Devices

ENCS 539

Project #4

Hungry Cat

Student Name: Mohammad Kabajah

ID: 1110600

Instructor's Name: Stephen Taylor

Section #:1

Date: 10/11/2014

Hungry Cat

In the hunger cat game used the given codes and modify them to make the application work as requested, I create one more class and create some methods to complete the work, and modify some of the given methods.

All the files are included in the archive, and here is the sections that I modified or created.

In the BoardView class I added constant value for the dog if it is in the room, I also add in the onDraw method the section to print the dog if it is in the room. I also added two flags to know if the player wins or loses the game so I can print on the screen you win or you lose, here is the part of the code that drawtext on screen

```
if(isLost){
    int size= R.dimen.textSize;

    String text = "You lose";
    paint.setTextSize(getResources().getDimensionPixelSize(size));
    paint.getTextBounds(text, 0, text.length(), rect2);
    paint.setColor(Color.BLACK);
    canvas.drawText("You lose", (d-rect2.width())/2, d/2, paint);
}
if(isWon){
    int size= R.dimen.textSize;
    String text = "You win";
    paint.setTextSize(getResources().getDimensionPixelSize(size));
    paint.getTextBounds(text, 0, text.length(), rect2);

    paint.setColor(Color.BLACK);
    canvas.drawText("You win", (d-rect2.width())/2, d/2, paint);
}
```

I also added a method to add the dog to the room when initialized

```
public void dog(Dog d ,Drawable x){
    dog = drawableToBitmap(x);
    rooms[d.row][d.col]|=dogHere;
}
```

In the Cal class I modified the partially Update Position method to check if the room we are going to contain a mouse it will be collected and we will add the collected mouse to the total number of collected mice by incrementing the counter which was defined in the PacCat Class.

```
if((board.rooms[row][col]&BoardView.mouseHere)!=0){
    board.rooms[row][col] &= (0xff ^BoardView.mouseHere); // the cat
    gets the mouse, if present
    PacCat.count++;
}
```

I created a new class Dog extends Drifter, in the Dog class it initialize the dog in random room in the maze, and contain the method partially update position, in this method we first remove the dog from the room was in and called the super method to change the room then add the value that the dog is in this room and here is the code of the class.

```

public class Dog extends Drifter{
    static Random random = new Random();
    public Dog(BoardView b) {
        super(b, random.nextInt(10), random.nextInt(10), Drifter.STUCK);
    }

    @Override
    void partiallyUpdatePosition(){
        board.rooms[row][col] &= (0xff ^ BoardView.dogHere);
        super.partiallyUpdatePosition();
        board.rooms[row][col] |= BoardView.dogHere;
    }
}

```

In the Drifter class I added a method PartiallyUpdatePosition() that is called from the cat and dog classes, and the method called the other PartiallyUpdatePosition(long now) method,

```

void partiallyUpdatePosition() {
    long now = System.currentTimeMillis();
    partiallyUpdatePosition(now);
}

```

In the PacCat class I created three objects of the dog class they will be the dogs that will catch the cat in the game, I modified the thread that was in the onCreate method that was updating the position of the cat and check if the cat and the dog are in the same room to finish the game, and if the counter of the collected mice is the total number of mice which 100 to end the game too, if the dog and the cat are in the same room the game will stop and the threads will stop working and we will change the flag that you lose to true and will print on the screen you lose, and in case of the wining all the threads will stop working and will change the flag that indicate you win to true and will print on the screen that you win.

Also I created three threads for the three dogs that will try to catch the cat in each thread first I make sure that the cat thread is still working before continue working in the dog thread, if it is not working then the thread will stop working, else it will keep working and all the three dogs work in the same way.

And here is the code for the cat and dog threads

```

tc =new Thread(){
    @Override
    public void run(){
        // run forever
        while (true) {
            //cat.updatePosition();
            cat.partiallyUpdatePosition();//Mohammad Kabbajah
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    boardView.invalidate();
                }
            });
            if((boardView.rooms[cat.row][cat.col]&BoardView.dogHere)!=0){
                PacCat.endGame();
                boardView.isLost=true;
                return;
            }
        }
    }
}

```

```

    }
    if(count == 100){
        PacCat.endGame();
        boardView.isWon=true;
        return;
    }
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            boardView.invalidate();
        }
    });
    try {
        sleep(500); // redraw 20 frames / sec
    } catch (Exception e){e.printStackTrace();} // log exception
and quit
    }
}
};
tc.start();
//dog1
t1 = new Thread(){
    @Override
    public void run(){

        while(true){
            if(tc==null){
                return;
            }
            dog1.partiallyUpdatePosition();
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    boardView.invalidate();
                }
            });
            try {
                sleep(500);
                if(dog1.direction==Drifter.STUCK)
                    dog1.direction=random.nextInt(4);
                else{
                    int r = dog1.row;
                    int c = dog1.col;
                    if((dog1.direction==0 || dog1.direction==2)
                        if(((boardView.rooms[r][c])&2)==0
|(((boardView.rooms[r][c])&8)==0 )
                            dog1.direction=random.nextInt(4);
                    if((dog1.direction==1 || dog1.direction==3)
                        if(((boardView.rooms[r][c])&1)==0
|(((boardView.rooms[r][c])&4)==0 )
                            dog1.direction=random.nextInt(4);
                    }
                } catch (Exception e){e.printStackTrace();} // log exception
and quit
            }
        }
    }
};

```

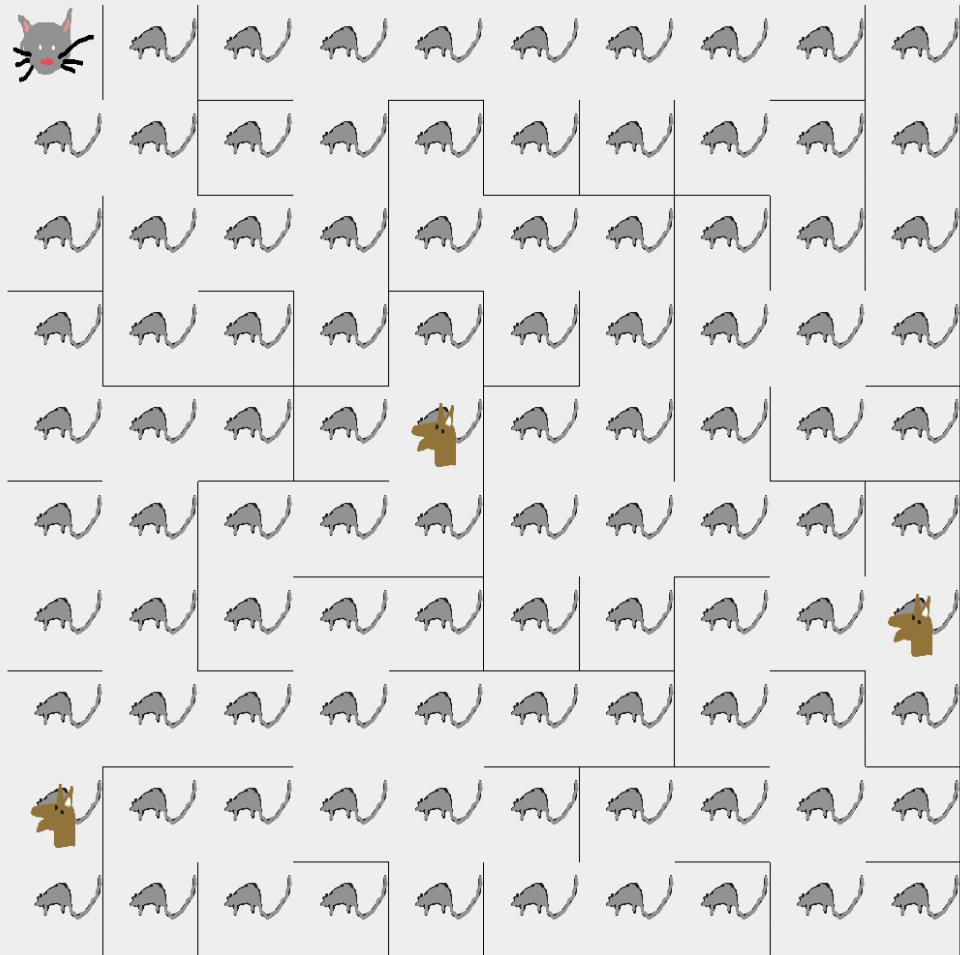
After creating all the dog threads we start them on after the other between each one 250 millisecond.


I also created a method called endgame, this method is used to stop the threads from working I tried the thread.stop() method but it can't be used, it crashes the application so using this we interrupt the thread then set it to null so there will be no thread to run, so the thread will stop working.

```
protected static void endGame() {  
    try{  
        tc.interrupt();  
        tc=null;  
        t1.interrupt();  
        t2.interrupt();  
        t3.interrupt();  
        t1=null;  
        t2=null;  
        t3=null;  
    }catch(Exception e){  
        e.printStackTrace();  
    }  
}
```

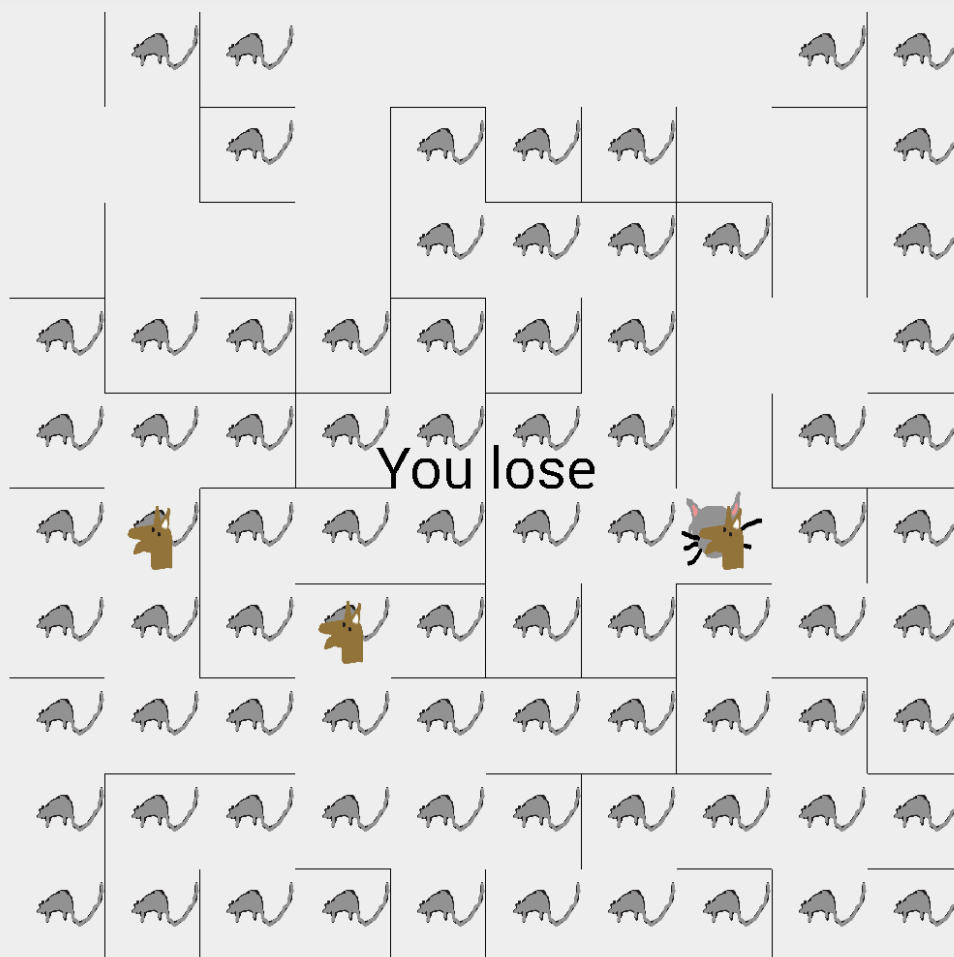
This is the last thing in the code and the application running and here is some screenshots of the game running on my phone.

Hungry Cat



 Saving screenshot...

Hungry Cat



Copied to clipboard.

<

^

V

>