

## Forside til eksamensprojekt

	OOPA/MGA-T-401									
	X									

Sæt X i rubrikken under dit fag:

*Underviser:* Jacob Nordfalk

*Eksamensform: Eksamensform kan ej vælges - er udelukkende individuel eksamen*

Enmandsprojekt: ☒ Gruppeprojekt ☐

Studienr: 988303 Navn: Allan Hirsch

Studienr: \_\_\_\_\_ Navn: \_\_\_\_\_

Titel Android Boulder Dash

Fortroligt ☐

Må anvendes i undervisningen ☐

*Dato:* 09-12-2010

<u>Den studerendes underskrift</u>	<u>Den studerendes underskrift – kun ved</u>
------------------------------------	--

Formål.....	2
Afgrænsning .....	2
Applikations model: .....	2
Beskrivelse af aktiviteter og disses funktionalitet.....	3
Hoved menu (BoulderDash.java): .....	3
Help (HelpActivity.java): .....	3
High Score (HighScoreActivity.java):.....	4
Game (GameActivity.java): .....	5
GameView.java .....	5
Spilbaner.....	6
ControlButton.java.....	7
ScoreBoard.....	8
Lyd.....	9
Vibration.....	9
HighScore (Skrivning til fil): .....	9
Tastatur .....	10
Test.....	10
Konklusion: .....	10
 Henvisninger og litteraturliste: .....	11
CD: Hvad er der på CD'en: .....	11
 Bilag: .....	11
KILDEKODE: .....	11
HighScoreActivity.java .....	11
BoulderDash.java .....	12
GameActivity.java .....	13
GameView.java .....	17
HelpActivity.java.....	31
ControlButton.java.....	32
game.xml.....	33
main.xml .....	34
strings.xml .....	35
AndroidManifest.xml.....	37
Boulderdash_help.html .....	37

## Formål:

Formålet med projektet er at lave en Android applikation, som gør brug af nogle af de emner gennemgået i kurset MGA (Mobil programmering på Google Android).

Som projekt har jeg valgt at portere det gamle Commordore64 spil, Boulder Dash, til Android, og anvende nogle af de spændende muligheder Android tilbyder i spillet.

Boulder Dash findes allerede som Java applet under open source på:

<http://javaboutique.internet.com/Boulderdash/> (Se reference liste).

Denne applet har jeg 'skrællet' for spil-logik som således bliver brugt i Android Applikationen.

Applikationen giver brugeren mulighed for at styre spilfiguren rundt, når der trykkes på spilkontrol knapperne. Spillet går i alt sin enkelthed ud på at navigere figuren rundt i miner og indsamle diamanter og undgå fjender. Undervejs samles der point, som både skrives til skærmen, men også til en highscore fil efter endt spil. Applikationen indeholder også diverse effekter, bl.a. lyd. Der implementeres også en hjælp funktionalitet, som forklarer spillet nærmere. Disse forskellige funktioner tilgås via en hovedmenu.

## Afgrænsning

Spillet er kun implementeret med to levels for overskuelighedens skyld. Men flere levels kan nemt tilføjes ved at lave flere <string-array>-elementer i strings.xml som forklaret i afsnit 'Spilbaner'.

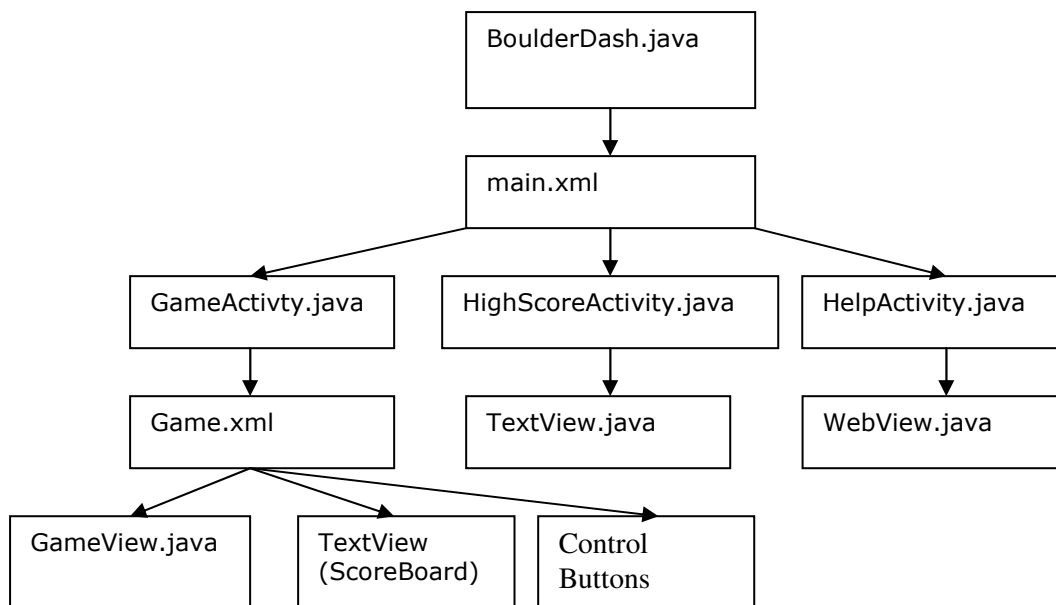
Spillet er implementeret uden mulighed for at pause og gemme et igangværende spil. Endvidere kan spillet kun spilles i portrait screenmode.

Ligeledes er der kun implementeret to lyde, da de kun skal vise multimediemulighederne for lyd i et spil på Android.

## Applikations model:

Android Applikationen er lavet som en app (applikation) med 4 aktiviteter. En hovedaktivitet (BoulderDash.java), som fungerer som hovedmenu og sikrer navigering til de 3 andre aktivitet i app'en:

- Hoved menu (BoulderDash.java)
  - HighScoreActivity
  - HelpActivity
  - GameActivity.



(Fig. 1: Applikations model)

To af aktiviteterne (**BoulderDash.java**, **GameActivity.java**) har contentviews, som er konfigureret via xml, hvorimod **highScoreActivity** og **HelpActivity** har contentviews, der konfigureres programatisk i Java kode. Dette gøres for at demonstrere begge metoder til håndtering af Viewgroups og views, og fordi contentviews i disse sidste klasse er simple.

### Beskrivelse af aktiviteter og disses funktionalitet

#### Hoved menu (**BoulderDash.java**):

Hovedaktiviteten består af et TextView samt 3 knapper, og fungerer som hovedmenu for applikationen.



Figur 2:Start menu (BoulderDash.java)

TextViewet angiver spillets titel og knapperne navigerer videre til app'ens andre aktiviteter ved at kalde med aktiviteternes respektive Intents i BoulderDash.onClick(). Komponenterne er grupperet i 2 LinearLayouts og et TableLayout, for at få stillet dem pænt op med luft til hinanden og til siderne (Se bilag: main.xml på side 34). Aktiviteterne registreres i Android manifest-filen.

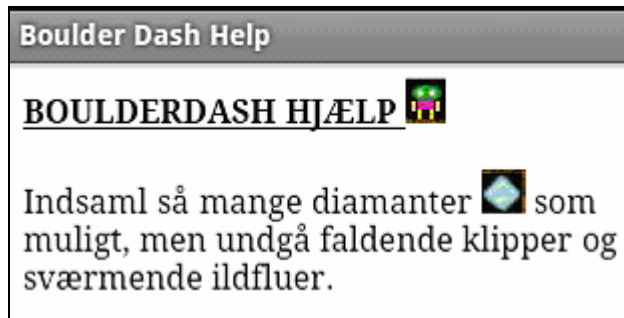
#### Help (**HelpActivity.java**):

Forklarer spillet til brugeren. Aktiviteten er implementeret med et WebView som contentview, fordi det gør det meget nemt at vedligeholde og præsentere hjælpdokumentet.

```
WebView help =new WebView(this);
help.loadUrl("file:///android_asset/BOULDERDASH_HELP.htm");
LinearLayout help_layout= new LinearLayout(this);
help_layout.addView(help,LayoutParams.FILL_PARENT,LayoutParams.FILL_PARENT);
this.setTitle("Boulder Dash Help");
setContentView(help_layout);
```

(Udsnit: onCreate() i HelpActivity.java)

Dokumentet kan således redigeres og vedligeholdes i et tekstredigeringsprogram, og herefter eksporteres som html dokument. Hermed kan man benytte styrkerne ved html til at præsentere hjælpen. Bl.a kan der vises billeder i dokumentet uden at skulle programmere en masse Java (se fig 3). Browserens scroll funktionalitet får man også med i købet.



**Figur 3: udsnit af hjælp dokument**

Html dokumentet placeres i Assets mappen, som er pakket ind i .apk filen, hvorfra Applikationen læser det.

Ulempen ved at bruge et WebView er, at det tager lidt tid at åbne siden, da det jo er en fuld browser, der startes op. Da hjælpen kun skal læses de første gange spillet spilles er dette acceptabelt.

### **High Score (HighScoreActivity.java):**



**Figur 4: udsnit af high score listen**

Her kan spilleren se de 5 højst opnåede scorere. Aktiviteten har et TextView som contentview. Denne aktivitets view er oprettet programmatisk i Java (se nedenfor) modsat de views, som er oprettet ved hjælp af xml.

```
TextView highscore=new TextView(this);
highscore.setText(readFile());//text is read here and inserted into textView
highscore.setGravity(Gravity.CENTER);
LinearLayout ll = new LinearLayout(this);
ll.addView(highscore,LayoutParams.FILL_PARENT,LayoutParams.WRAP_CONTENT);
setContentView(ll);
```

(Udsnit fra onCreate() i HighScoreActivity.java)

Android understøtter almindelig Java fil håndtering, så data læses fra en fil lokalt på telefonen ved først at finde mappen, dernæst filen - hvorefter der læses med en BufferedReader, der jo tillader at læse hele linier ad gangen.

```
File privatMappe = this.getFilesDir();
theFile=new File(privatMappe,"HighScore.txt");
BufferedReader reader = new BufferedReader(new FileReader(theFile));
String line= "";
String tmp="";
while((line=reader.readLine())!=null){
    tmp=tmp+line+"\n";
}
reader.close();
return tmp; }
```

(Udsnit fra readFile() i HighScoreActivity.java)

Filens indhold returneres til TextViewet (highscore) i HighScoreAktivitet.java som det præsenteres i fig 4.

Skrivning til filen foregår i Game Aktiviteten.

### Game (GameActivity.java):

Denne aktivitet er hjertet i app'en. Grafikken renderes her, lydene spilles, spil info opdateres og spillet kontrolleres her ved tast på spilknapperne.

ContentViewet er bygget op ved hjælp af forskellige viewgroups styret i game.xml (Bilag s.33). Yderst ligger et FrameView, som stabler de overliggende views på hinanden, hvilket gør at der kan lægges et view med gennemsigtig baggrund oven på selve spil-viewet. Dette gennemsigtige view indeholder et TextView, hvor forskellige spilinformationer skrives, bl.a. score, og i bunden af dette gennemsigtige view ses spilknapperne, som selv er gjort halv-gennemsigtige, så de ikke blokerer for grafikken i det underliggende lag.



Figur 5: GameActivity med Views i to lag

### GameView.java

Denne klasse sørger for at tegne grafikken.

Spillet er som sagt fundet som open source på nettet, så selve spillogikken er ikke noget jeg selv har kodet, men blot tilrettet så det kan køre på Android.

Et spil har ofte brug for at udnytte den fulde skærm. Derfor starter jeg med at fjerne titleBar og notification bar. Dette gøres i aktivitetens (GameActivity) onCreate() ved at give window manageren nogle Layout parametre, der fortæller, at jeg ønsker at bruge hele skærmen:

```
requestWindowFeature(Window.FEATURE_NO_TITLE);
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
    WindowManager.LayoutParams.FLAG_FULLSCREEN);
```

(onCreate() i GameActivity.java)



Tegnene i array'et oversættes til de rette bitmap-bidder ved hjælp af streng-værdien:

```
private static final String imgRep = ".:#WMPO+*BFA][R";  
(Erklæring af variabel i GameView.java)
```

Ud fra index-tal i bitmap-Array'et kan Android nu opdatere canvas.

```
if (imgRep.charAt(index) == levelMap[x][y]) {  
    canvas.drawBitmap(img[index], (x * SQUARE_DIMEN) - shiftViewX,  
    (y * SQUARE_DIMEN) + 40, null);
```

(Udsnit: onDraw() i GameView.java)

Variablen **shiftViewX** bruges til at flytte grafikken, når spilleren bevæger sig længst ud til højre i spillet. Spillet er nemlig for stort til at være på skærmen i x-aksen, så det undersøges hvor stor skærmen er og spilgrafikken flyttes **shiftViewX** til venstre. Denne beregning foretages i kontrollen af spilleren:

```
public boolean controls(int controlID, int scrWidth, int scrHeight)  
(controls() i GameView.java)
```

Udover spilfiguren har spillet også nogle skurke, såkaldte bots. Dem skal man undgå, og de styres af deres egen tråd. Den startes i GameActivity (botsThread), men er implementeret som indre klasse (GraphicsThread) i GameView.java, da tråden bruger lokale variabler og objekter fra denne klasse.

Tråden er lavet til at holde styr på skurkene (Bots) og til at beregne bevægelser for spiller og faldende sten og diamanter, og det er her at **levelMap**-array'et vedligeholdes, det vil sige opdateres med billede koordinater.

Til at gøre alt dette, gør koden brug af forskellige lister og koordinat objekter, som jeg dog ikke vil omtale yderligere, da det er kode jeg har lånt fra open source. Dog skal det nævnes at postInvalidate() også kaldes herfra, og derfor er med til at animere grafikken.

Denne tråd har også vist sig at skabe lidt problemer i forhold til at gemme spillet i forbindelse med aktivitets livscyklus (onCreate, onResume, onPause() etc.). Dette skyldes problemer med at synkronisere objekter, der gemmer spil-data, bl.a det meget omtalte **levelMap** array.

Derfor er spillet implementeret uden mulighed for at pause og gemme spillet. Endvidere kan spillet kun spilles i portrait screenmode, da en vending af telefonen også vil kalde disse livscyklusmetoder.

Skærmens orientering låses ved at tilføje nedenstående i manifest-filen:

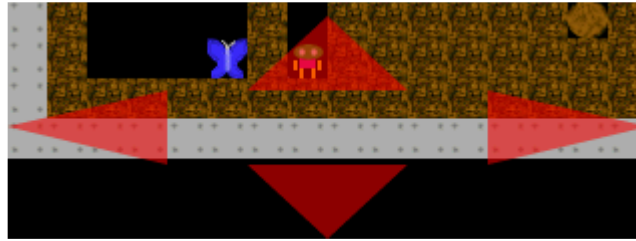
```
<activity android:name="org.games.boulderdash.GameActivity"  
    android:screenOrientation="portrait">
```

De andre aktiviteter kan dog godt læses i landscape mode.

### ControlButton.java

For at implementere spilknapperne har jeg lavet min egen knapklasse ControlButton.java (Bilag s.32), nedarvet fra Button. Hver knap er udstyret med ID i ressource filen og dette ID smager jeg på i controlButton's onDraw metode, og tegner så knappen i den rigtige retning ved hjælp af et path og paint object. Knappens Alpha-værdi sættes til 150 for at gøre den let gennemsigtig, så den kan placeres ovenpå spil-viewet. Nedenfor ses spil-figuren igennem én af knapperne (fig. 7).





**Figur 7: gennemesigtige knapper ovenpå gameview'et**

I Game aktiviteten tilføjes knapperne en onClick lytter, som håndterer brugerens skærmtryk:

```
private void setListeners() {
    for (int i = 0; i < keys.length; i++) {
        final int t = i ;
        keys[i].setOnClickListener(new View.OnClickListener(){
            public void onClick(View v) {
                theGame.controls(keys[t].getId(),width, height);
                scoreBoard.setText(theGame.updateScore());
            }
        });
    }
}
```

(setListeners() i GameActivity.java)

Knapperne tilføjes deres lyttere generisk, da det jo stort set er det samme de skal gøre. onClick kalder controls() i GameView.java. Her beregnes retning og afstand for spilfiguren, og postinvalidate() kaldes for at tegne spillet igen. I controls() beregnes også om spilfiguren er kommet så langt til højre i banen, at denne del nu skal tegnes. Derfor kaldes controls() med skærmens vidde (width), så variabelen **shiftViewX** kan beregnes, når onDraw() kaldes i GameView.java. Det vil så tegne den del af banen, der ikke er synlig.

```
//move screen to right-side mode
if(move.x>=(scrWidth/20)){
    shiftViewX=400-scrWidth; //banen er 400 pics, da det er en 20*20 matrix
}
//move screen back to initail mode
if(move.x<=(20-(scrWidth/20))&&shiftViewX!=0){
    shiftViewX=0;
}
```

(fra controls() i GameView.java)

### ScoreBoard

Scoren vedligeholdes også i denne metode (controls()), samt andre spilinformationer som brugeren kan se i TextView'et øverst i skærmen:

**Figur 8: ScoreBoard**

## Lyd

Der er også implementeret to lyde. En lyd, når spilfiguren dør og en anden lyd, når han samler en diamant op.

Hver lyd har sin egen mediaplayer, som konstrueres, når Game aktiviteten starter op.

```
diamondMP=MediaPlayer.create(this, R.raw.diamond_sound);  
diamondMP.setVolume(0.5f, 0.5f);
```

(Udsnit af onCreate() i GameActivity)

Lydene kaldes henholdsvis fra controls() og run (), da det er her viden om de hændelser, som kræver lyd, opstår.

Dette gøres ved at kalde playSound() i GameActivity med ressource id for lyden som parameter.

Der ryddes op i livscyklus-metode onDestroy(), dvs at ressourcerne (mp3 lydene) frigives.

## Vibration

Desuden er der også implementeret en vibrationseffekt, når spilleren dør.

```
Vibrator v = (Vibrator) parent.getSystemService(Context.VIBRATOR_SERVICE);  
v.vibrate(300);
```

(Udsnit af run() i GameView.java)

At få app'en til at vibrere kræver, at der gives tilladelse til dette i Android manifestet:

```
<uses-permission android:name="android.permission.VIBRATE"/>
```

(Udsnit af AndroidManifest.xml)

## HighScore (Skrivning til fil):

Når spillet lukkes lagres highscoren lokalt i en text fil:  
/data/data/org.games.boulderdash/Highscore.txt

Data sorteres og lagres i nedstigende rækkefølge i filen. Dette gøres ved at lagre data midlertidigt i en ArrayList, som så kan sorteres ved hjælp af statiske metoder i Collections klassen:

```
if(newScore>0)  
    ar.add(newScore);//Add score  
Collections.sort(ar);//sorting the collection  
Collections.reverse(ar);//Descending order  
  
if(ar.size()>5)//only 5 scores  
    ar.remove(ar.size() - 1);//so remove last one  
  
if(fw==null)  
    fw = new FileWriter(theFile);
```

(Udsnit af storeHighScore() i GameActivity.java)

Derefter skrives data ind i filen v.hj.a en filWriter. Forlades spillet i utide kaldes denne metode fra livcyklus-metode onPause().

```
@Override  
protected void onPause() {  
    super.onPause();  
    try {  
        storeHighscore(theGame.getScore());
```

(onPause() i GameActivity.java)

Spillet kan enten lukkes ved at brugeren selv forlader aktiviteten, eller at spilfiguren dør. Dør figuren kaldes finish() på aktiviteten, som så gennemløber livcyklus metoderne og lukker aktiviteten. Det vil så være den foregående aktivitet, BoulderDash, som vises.

```
if (playerDead) {  
    //give a little pause before returning  
    try {  
        Thread.sleep(2000);  
    } catch (InterruptedException ex) {  
        Logger.getLogger(GameView.class.getName()).log(Level.SEVERE, null, ex);  
    }  
    parent.finish();//kill activity and return to menu.  
}
```

(Udsnit fra run() i GameView)

## Tastatur

Spillet kan også styres ved hjælp af tastaturknapper, hvis sådan nogle er tilgængelige. Ellers bruges denne feature, når spillet testes i emulatoren. Dette er gjort muligt ved at registrere en onKeyDown() lytter i GameActivity.java. Her smages på om det er en af pile-tasterne der er trykket på, og hvis det er tilfældet kaldes videre til gameView.controls() med knap ID.

## Test

Applikationen er testet på:

- Emulatoren. Android version. 2.2
- Android telefon:
  - HTC HERO med Android 2.1
  - Sony Ericsson X10 Mini Pro med Android 2.1
- Applikationen er testet ved at gennemføre spillet, og se at scoren bliver vist og beregnet, og anden spilinformation bliver opdateret.
- Spilleren bevæger sig som forventet, når der trykkes på spilknapperne.
- Grafikken flyttes i x-planet, når spilleren bevæger sig over i den usynlige del af spillet.
- Lydene gengives som de skal når der samles diamanter og når spillet slutter ved spilfigurs død.
- Ligeledes vibrerer telefonen også ved figurens død.
- Highscore-listen opdateres korrekt og viser de fem seneste scorer.
- Hjælp funktionaliteten virker også efter hensigten.
- Spillet bliver i portrait mode når telefonen vendes.
- Emulatorens pile-taster virker også korrekt i forhold til navigering af spilfigur.

## Konklusion:

Rapportens formål er nået. Boulder Dash kan nu spilles på Android platformen.

Applikationen giver brugeren mulighed for at styre spilfiguren rundt, når der trykkes på spilkontrol knapperne. Der samles point, som både skrives til skærmen og til en highscore fil efter endt spil.

Applikationen indeholder lyd og vibrerings- effekter.

Hjælp funktionalitet, som forklarer spillet nærmere, er implementeret.

De forskellige funktioner tilgås via en hovedmenu.

**Henvisninger og litteraturliste:**

Bøger: "Hello Android", Ed Burnette.  
Kildekode: "BOULDERDASH", Dave Koelle  
<http://javaboutique.internet.com/Boulderdash/>

**CD: Hvad er der på CD'en:**

BoulderDash.apk.  
Rapporten.  
Boulderdash.zip (boulderdash som java applet).  
BoulderDash netbeans projekt.

**Bilag:****KILDEKODE:****HighScoreActivity.java**

```
package org.games.boulderdash;

import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.WindowManager.LayoutParams;
import android.widget.LinearLayout;
import android.widget.TextView;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author AH
 */
public class HighScoreActivity extends Activity {

    private File theFile;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        try {

            this.setTitle("HIGH SCORE");
            TextView highscore=new TextView(this);
            highscore.setText(readFile());
            highscore.setPadding(0, 100, 0,0);
            highscore.setGravity(Gravity.CENTER);
            highscore.setTextSize(30);
            LinearLayout ll = new LinearLayout(this);
            ll.addView(highscore,LayoutParams.FILL_PARENT,LayoutParams.WRAP_CONTENT);
            setContentView(ll);
```

```

    } catch (FileNotFoundException ex) {
        Logger.getLogger(HighScoreActivity.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(HighScoreActivity.class.getName()).log(Level.SEVERE, null, ex);
    }
}
public String readFile()throws FileNotFoundException, IOException{

    File privateFolder = this.getFilesDir();//data/data/pakkenavn/filnavn

    theFile=new File(privateFolder,"HighScore.txt");

    BufferedReader reader = new BufferedReader(new FileReader(theFile));

    String line= "";

    String tmp="";
    while((line=reader.readLine())!=null){
        tmp=tmp+line+"\n";
    }
    reader.close();

    return tmp;

}

}

```

### **BoulderDash.java**

```

package org.games.boulderdash;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

/**
 *
 * @author AH
 */
public class BoulderDash extends Activity implements OnClickListener{

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle icle) {
        super.onCreate(icle);
        setContentView(R.layout.main);

        Button game=(Button) findViewById(R.id.new_game);
        game.setOnClickListener(this);

        Button help=(Button) findViewById(R.id.help_button);
        help.setOnClickListener(this);
    }
}

```

```

        Button highScore=(Button) findViewById(R.id.highscore_button);
        highScore.setOnClickListener(this);

    }

    /*
    Hovedmenu : navigering til andre funksjoner i app'en
    */

    public void onClick(View button) {

        Intent i=null;

        switch(button.getId()){

            case R.id.help_button:
                i =new Intent(this, HelpActivity.class);

                break;

            case R.id.new_game:
                i =new Intent(this, GameActivity.class);

                break;
            case R.id.highscore_button:
                i =new Intent(this, HighScoreActivity.class);

                break;
        }
        if(i!=null)
            startActivity(i);
    }

}

```

### **GameActivity.java**

```

package org.games.boulderdash;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.Display;
import android.view.KeyEvent;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.TextView;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Collections;

```

```

import java.util.logging.Level;
import java.util.logging.Logger;
import org.games.boulderdash.GameView.GraphicsThread;

/**
 *
 * @author AH
 */
public class GameActivity extends Activity {

    private static final String TAG= "BoulderDash";

    private final View keys[] = new View[4];
    private GameView theGame;
    public TextView scoreBoard;
    private int width;
    private int height;
    MediaPlayer diamondMP;
    MediaPlayer explosionMP;

    GraphicsThread botThread;//Thread controlling bots and such

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle icle) {

        super.onCreate(icle);

        diamondMP=MediaPlayer.create(this, R.raw.diamond_sound);
        diamondMP.setVolume(0.5f, 0.5f);

        explosionMP=MediaPlayer.create(this, R.raw.explosion_sound);
        explosionMP.setVolume(0.5f, 0.5f);

        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                               WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.game);

        theGame =(GameView)findViewById(R.id.gameView);

        //textview øverst i skærmen der fungerer spil display
        scoreBoard=(TextView)findViewById(R.id.score_board);
        this.scoreBoard.setText(theGame.updateScore());

        //til at beregne hvornår spilen er uden for skærmen
        Display display = getWindowManager().getDefaultDisplay();
        width = display.getWidth();
        height = display.getHeight();

        findViews(); //finding control buttons
        setListeners();

    }
    //til at håndtere tastatur tryk
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        boolean bool= super.onKeyDown(keyCode, event);
        int direction=0;
    }

```

```

switch(keyCode){
    case KeyEvent.KEYCODE_DPAD_UP:
        direction=R.id.up;
        break;
    case KeyEvent.KEYCODE_DPAD_DOWN:
        direction=R.id.down;
        break;
    case KeyEvent.KEYCODE_DPAD_LEFT:
        direction=R.id.left;
        break;
    case KeyEvent.KEYCODE_DPAD_RIGHT:
        direction=R.id.right;
        break;

    default:
        return bool;
}

//delegerer videre til skærmknapperne
theGame.controls(direction, width, height);

//opdater spilinfo
scoreBoard.setText(theGame.updateScore());
return true;
}

void playSound(int SoundId){
    switch(SoundId){
        case R.raw.diamond_sound:
            diamondMP.start();
            break;
        case R.raw.explosion_sound:
            explosionMP.start();
            break;
    }
}

@Override
protected void onPause() {
    super.onPause();
    try {
        storeHighscore(theGame.getScore());
        botThread.setActive(false);//releasing runnable
    } catch (IOException ex) {
        Logger.getLogger(GameActivity.class.getName()).log(Level.SEVERE, null, ex);
    }
}

@Override
protected void onDestroy() {
    super.onDestroy();

    diamondMP.release();//releasing resources
    explosionMP.release();
}

```



```

@Override
protected void onResume() {
    super.onResume();

    botThread = theGame.newThread();
    botThread.start();
}

```

```

private void findViews() {
    keys[0]= findViewById(R.id.up);
    keys[1] = findViewById(R.id.right);
    keys[2] = findViewById(R.id.left);
    keys[3] = findViewById(R.id.down);
}

```

```

}

```

```

private void setListeners() {
    for (int i = 0; i < keys.length; i++) {
        final int t = i ;

        keys[i].setOnClickListener(new View.OnClickListener(){
            public void onClick(View v) {

                theGame.controls(keys[t].getId(),width, height);
                scoreBoard.setText(theGame.updateScore());

            }
        });
    }
}

```

```

public void storeHighscore(String theScore) throws IOException {

    BufferedReader reader = null;
    FileWriter fw=null ;
    File privatMappe = this.getFilesDir();
    File theFile = new File(privatMappe, "HighScore.txt");

    try {
        reader = new BufferedReader(new FileReader(theFile));
    } catch (FileNotFoundException ex) {
        fw = new FileWriter(theFile);//if no file, make one
        Logger.getLogger(GameActivity.class.getName()).log(Level.SEVERE, null, ex);
    }

    String tmp= "";
    String line = "";
    ArrayList<Integer> ar = new ArrayList<Integer>();

    if(reader!=null){
        while ((line = reader.readLine()) != null) {

```

```

        ar.add(Integer.parseInt(line));
    }
    reader.close();
}

int newScore=Integer.parseInt(theScore);

if(newScore>0)
    ar.add(newScore);//Add score

Collections.sort(ar);//sorting the collection
Collections.reverse(ar);//Descending order

if(ar.size()>5)//only 5 scores
    ar.remove(ar.size() - 1);//so remove last one

if(fw==null)
    fw = new FileWriter(theFile);

PrintWriter pw = new PrintWriter(fw);
for (Integer val : ar) {for each
    pw.println(val);
}
fw.close();
}
}

```

### **GameView.java**

```

package org.games.boulderdash;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics_PICTURE;
import android.view.MotionEvent;
import android.view.View;
import android.graphics.drawable.Drawable;
import android.os.Vibrator;
import android.util.AttributeSet;
import android.util.Log;
import java.io.IOException;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

```

```

/**
 *
 * @author AH
 */
public class GameView extends View {

    private final View keys[] = new View[1];
    private float shiftViewX=0;

```

```

private float shiftViewY=0;
boolean isAlive;
static boolean runThread;

ArrayList<Coord> redraw = new ArrayList<Coord>();
//ArrayList sounds = new ArrayList();
private static final int SLEEP_DELAY = 1750;
int maxtimer=1,timer,level;
int score,accumulatedScore,scorePerDiamond=5,scorePerExtraDiamond=10;

public static final int UP    = 0;
public static final int RIGHT = 1;
public static final int DOWN  = 2;
public static final int LEFT  = 3;

// Game variables
Coord coordPlayer,coordExit;
int numNeededDiamonds,numFulfilledDiamonds;
boolean levelDone,playerAlmostDead,playerDead;

// Arrays to handle input from APLET tag
String[] arrNumNeededDiamonds,arrScorePerDiamond,arrScorePerExtraDiamond;
int[] arrMaxtimer;
String[] arrLevelTitle,arrLevelAuthor,arrImageFilename;

// Image and icon data
private Bitmap[] img;
private static final int NUM_IMAGES = 15;
private static final String imgRep = ".:#WMPO+*BFA][R";//string to set up map
private String imageFilename = "images.gif";
private static int[][] imgCoord =
{
    { 0, 0,20,20}, // Space
    { 20, 0,20,20}, // Dirt
    { 40, 0,20,20}, // Titanium Wall
    { 0,20,20,20}, // Brick Wall
    { 20,20,20,20}, // Magic Wall
    { 40,20,20,20}, // Player
    { 0,40,20,20}, // Rock
    { 20,40,20,20}, // Diamond
    { 40,40,20,20}, // Explosion
    { 0,60,20,20}, // Butterfly
    { 20,60,20,20}, // Firefly
    { 40,60,20,20}, // Amoeba
    { 0,80,20,20}, // Closed Exit
    { 20,80,20,20}, // Open Exit
    { 40,80,20,20}}; // Robot (not programmed)

// Map and level data
private static final int X_DIMEN = 20;
private static final int Y_DIMEN = 20;
private static float SQUARE_DIMEN = 20;
private char[][] levelMap;
private int NUM_LEVELS;
private String[][] levelData;
ArrayList <BotPosition> bots ;
Drawable theImage;
GameActivity parent;

```

```

public String updateScore(){
    String scoreBoardTxt="Score: "+score+" ";
    scoreBoardTxt=scoreBoardTxt+"Level: "+level+" / "+NUM_LEVELS;
    scoreBoardTxt=scoreBoardTxt+"\nDiamonds: "+numFulfilledDiamonds+" /
"+(numFulfilledDiamonds<numNeededDiamonds ? numNeededDiamonds:0);
    scoreBoardTxt=scoreBoardTxt+"   Diamond Points: "+(numFulfilledDiamonds<numNeededDiamonds
? scorePerDiamond : scorePerExtraDiamond);

    return scoreBoardTxt;

}

public GameView(Context context, AttributeSet as) {
    super(context,as);
    parent=((GameActivity)context);
    loadImages();

    // Get parameters from the HTML file
    String numlevels ="2";// getResources().getString( R.string.levels);

    if (numlevels == null) return;
    NUM_LEVELS = Integer.parseInt(numlevels);
    levelData = new String[NUM_LEVELS][Y_DIMEN];

    arrNumNeededDiamonds = getResources().getStringArray(R.array.NEEDED_DIAMONDS);
    arrScorePerDiamond = getResources().getStringArray(R.array.POINTS_PER_DIAMOND);
    arrScorePerExtraDiamond =
getResources().getStringArray(R.array.POINTS_PER_EXTRA_DIAMOND);

    arrMaxtimer = new int[NUM_LEVELS];
    arrLevelTitle = new String[NUM_LEVELS];
    arrLevelAuthor = new String[NUM_LEVELS];
    arrImageFilename = new String[NUM_LEVELS];

    int[]level_mapping={R.array.level_data_1,R.array.level_data_2};

    for (int i=0; i<NUM_LEVELS; i++) {
        for (int u=0; u<Y_DIMEN; u++) {
            String level_Data[]=getResources().getStringArray(level_mapping[i]);

            levelData[i][u] =level_Data[u];

        }

        arrLevelTitle[i] = getResources().getString(R.string.LEVEL_TITLE_1);
        if (arrLevelTitle[i] == null) arrLevelTitle[i] = "Unnamed";
        arrLevelAuthor[i] = getResources().getString(R.string.LEVEL_AUTHOR_1);
        if (arrLevelAuthor[i] == null) arrLevelAuthor[i] = "Anonymous";
        arrImageFilename[i] = getResources().getString(R.string.IMAGEFILE_1);
        if (arrImageFilename[i] == null)
            arrImageFilename[i] = "images.gif";

    }
}

```

```

// Set up player variables
coordPlayer = new Coord();
coordExit = new Coord();

// Just in case someone forgets to define these, let's make them exist
coordPlayer.x = 2; coordPlayer.y = 2;
coordExit.x = 1; coordExit.y = 1;

// Set up the map
if(levelMap==null){
    levelMap = new char[X_DIMEN][];
    for (int i=0; i<X_DIMEN; i++) {
        levelMap[i] = new char[Y_DIMEN];
    }
}
else
    isAlive=true;

level = 1;
setupLevel(level);

}

private void setupLevel(int level) {

/*
if(parent.scoreBoard!=null)
    parent.scoreBoard.setText(updateScore());
*/
numNeededDiamonds = Integer.parseInt(arrNumNeededDiamonds[level-1]);
scorePerDiamond = Integer.parseInt(arrScorePerDiamond[level-1]);
scorePerExtraDiamond = Integer.parseInt(arrScorePerExtraDiamond[level-1]);

/* Set up the level */
shiftViewX=0;
levelDone = false;
playerAlmostDead = false;
playerDead = false;
bots = new ArrayList<BotPosition>();
numFulfilledDiamonds = 0;
for (int y=0; y<Y_DIMEN; y++) {
    for (int x=0; x<X_DIMEN; x++) {

        levelMap[x][y] = levelData[level-1][y].charAt(x);

        drawCoord(x,y);
        if ((levelMap[x][y] == '[') || (levelMap[x][y] == ']')) {
            coordExit.x = x;
            coordExit.y = y;
        } else
            if (levelMap[x][y] == 'P') {
                coordPlayer.x = x;
                coordPlayer.y = y;
            } else
                if ((levelMap[x][y] == 'B') || (levelMap[x][y] == 'F')) {
                    int dir = (levelMap[x][y] == 'F')?0:1;

```

```

        BotPosition botpos = new BotPosition(x,y,1,dir);
        bots.add(botpos);
    }

}

}

}

private void loadImages()
{
    // Break up source image into individual images
    // First, load the large image
    Bitmap sourceImage = BitmapFactory.decodeStream(getResources().openRawResource(
        R.drawable.images));

    // Now break it up into pieces
    img = new Bitmap[NUM_IMAGES];
    for (int i=0; i<NUM_IMAGES; i++) {
        img[i] = cropSourceImage(sourceImage,imgCoord[i]);
    }
}

private Bitmap cropSourceImage(Bitmap sImage, int[] iCoord){

    //then Crop the obtained 'bitmap'
    Bitmap croppedBitmap= Bitmap.createBitmap(sImage,iCoord[0],iCoord[1],iCoord[2],iCoord[3]);
    return croppedBitmap;

}

@Override
protected void onDraw(Canvas canvas) {

    for (int x = 0; x < X_DIMEN; x++) {
        for (int y = 0; y < Y_DIMEN; y++) {
            int index;
            for (index = 0; index < imgRep.length(); index++) {
                if (imgRep.charAt(index) == levelMap[x][y]) {
                    canvas.drawBitmap(img[index], (x * SQUARE_DIMEN) - shiftViewX, (y *
SQUARE_DIMEN) + 40, null);
                }
            }
        }
    }
}

public boolean controls(int controlID, int scrWidth, int scrHeight) {

    if (playerAlmostDead) {
        return false;
    }
}

```

```

Coord delta = new Coord();
Coord delta2 = new Coord();
Coord move = new Coord();
Coord move2 = new Coord();
boolean validMove = false;

// Interpret the keycode
if (controlID == R.id.up) {
    delta.x = 0; delta.y = -1;
} else
if (controlID == R.id.right) {
    delta.x = 1; delta.y = 0;
} else
if (controlID == R.id.down) {
    delta.x = 0; delta.y = 1;
} else
if (controlID == R.id.left) {
    delta.x = -1; delta.y = 0;
}

// Make move relative to player
move.x = coordPlayer.x + delta.x;
move.y = coordPlayer.y + delta.y;

//move screen to right-side mode
if(move.x>=(scrWidth/20)){
    shiftViewX=400-scrWidth;
}
//move screen back to initail mode
if(move.x<=(20-(scrWidth/20))&&shiftViewX!=0){
    shiftViewX=0;
}

// Range checking on the move
if (move.x < 0) { move.x = 0; } else
if (move.x > X_DIMEN) { move.x = X_DIMEN; }
if (move.y < 0) { move.y = 0; } else
if (move.y > Y_DIMEN) { move.y = Y_DIMEN; }

// Create a "look-ahead" position
delta2.x = delta.x*2;
delta2.y = delta.y*2;
move2.x = coordPlayer.x + delta2.x;
move2.y = coordPlayer.y + delta2.y;

// Moving into an empty space? "I'll allow it!" -- Mills Lane
if ((levelMap[move.x][move.y] == '.') || (levelMap[move.x][move.y] == ':')) {
    if (levelMap[move.x][move.y] == ':') {
        redraw(move); // SPACE sound
    } else {
        redraw(move); // DIRT sound
    }
    validMove = true;
} else

// Pushing a rock?
if ((levelMap[move.x][move.y] == 'O') && (move2.y != move.y-1)) {
    if (((move2.x >= 0) && (move2.x <= X_DIMEN)) &&

```

```

        ((move2.y >= 0) && (move2.y <= Y_DIMEN))) {
    if (levelMap[move2.x][move2.y] == '.') {
        // "look-ahead" is an empty space. Move the rock.
        levelMap[move2.x][move2.y] = levelMap[move.x][move.y];
        redraw(move2);
        redraw(move);
        validMove = true;
    }
}
} else

// Grabbing a Diamond?
// Add one to the Number of Fulfilled Diamonds,
// and see if the exit should be opened
if (levelMap[move.x][move.y] == '+') {
    numFulfilledDiamonds++;
    score += (numFulfilledDiamonds < numNeededDiamonds ? scorePerDiamond :
scorePerExtraDiamond);
    updateScore();
    redraw(move);
    parent.playSound(R.raw.diamond_sound);
    if (numFulfilledDiamonds == numNeededDiamonds) {
        levelMap[coordExit.x][coordExit.y] = '[';
        redraw(coordExit);
    }
    validMove = true;
} else

// If moving into an exit, make sure it's open!
if (levelMap[move.x][move.y] == '[') {
    redraw(move);
    validMove = true;
}

if (validMove) {
    levelMap[move.x][move.y] = 'P';
    levelMap[coordPlayer.x][coordPlayer.y] = '.';
    drawCoord(coordPlayer.x, coordPlayer.y);
    coordPlayer.x = move.x;
    coordPlayer.y = move.y;
}

// Redraw coordinates that need to be redrawn
redrawCoords();

// Is the level over?
if ((coordPlayer.x == coordExit.x) &&
    (coordPlayer.y == coordExit.y)) {

    levelDone = true;
    // All done with the level! Give a little pause...
    try {
        Thread.sleep(SLEEP_DELAY);
    } catch (Exception e) {
        Log.e("CONTROLS()", e.getMessage());
    }

    accumulatedScore = score;
    if (level+1 < NUM_LEVELS+1) {
        level++;
    }
}

```



```

    } else {
        level = 1;
    }
    setupLevel(level);
}

return true;
}

@Override
public boolean onTouchEvent(MotionEvent event) {
    if (event.getAction() != MotionEvent.ACTION_DOWN)
        return super.onTouchEvent(event);

    // SQUARE_DIMEN = - event.getX();
    //y = - event.getY()/SQUARE_DIMEN;
    Log.d("onTouchEvent", "onTouchEvent: x " + event.getX() + "onTouchEvent: y " + event.getY());
    // postInvalidate();
    return true;
}

public void drawCoord(int x, int y)
{
    int index;
    for (index=0; index<imgRep.length(); index++) {
        if (imgRep.charAt(index) == levelMap[x][y]) {
            postInvalidate();//redraw canvas
        }
    }
}

public void redrawCoords() {
    // Redraw any coordinates that should be redrawn
    try {
        for (int i=0; i<redraw.size(); i++) {
            Coord display = (Coord)redraw.get(i);
            drawCoord(display.x,display.y);
        }
    } catch (Exception e) {
        Log.e("REDRAWCOORDS()", e.getMessage());
    }
    redraw.clear();
}

public void redraw(Coord coord) {
    redraw.add(coord);
}

```

```

GraphicsThread newThread() {
    return new GraphicsThread();
}

public String getScore(){
    return score+""; //implicit cast to String
}

public class GraphicsThread extends Thread {
    public boolean active=true;

    public GraphicsThread(){
        super("Graphics thread");
    }

    public void setActive(boolean bool){

        active=bool;

    }

    @Override
    public void run() {

        // This will take care of animation

        boolean botSmash,rockSmash,amoebaSmash,explosionsQuenched=false;
        Coord smashCoord = new Coord();
        char me = '.';

        while (this.isAlive() && active) {
            if (playerAlmostDead && explosionsQuenched) {
                for (int y=Y_DIMEN-2; (y>=0) && (explosionsQuenched); y--) {
                    for (int x=0; (x<X_DIMEN) && (explosionsQuenched); x++) {
                        if (levelMap[x][y] == '*') {
                            explosionsQuenched = false;
                        }
                    }
                }
            }
            if (explosionsQuenched) {
                playerAlmostDead = false;
                playerDead = true;
            }
        }

        explosionsQuenched = false;
        for (int y=Y_DIMEN-2; y>=0; y--) {
            for (int x=0; x<X_DIMEN; x++) {
                botSmash = false;
                rockSmash = false;
                amoebaSmash = false;

                // Is this item smashed by amoeba?
                if (((y-1 > 0) && (levelMap[x][y-1] == 'A')) ||
                    ((x+1 < X_DIMEN) && (levelMap[x+1][y] == 'A')) ||
                    ((y+1 < Y_DIMEN) && (levelMap[x][y+1] == 'A')) ||
                    ((x-1 > 0) && (levelMap[x-1][y] == 'A')) &&
                    ((levelMap[x][y] == 'B') || (levelMap[x][y] == 'F')))) {

```

```

        amoebaSmash = true;
        smashCoord = new Coord(x,y);
    }

    // Quench all explosions
    if (levelMap[x][y] == '*') {
        levelMap[x][y] = '.';
        redraw(new Coord(x,y));
        explosionsQuenched = true;
    } else

// If, for any reason, the player is dead, end Game
// Do this AFTER the explosions take place
if (playerDead) {

    //give a little pause before returning
    try {
        Thread.sleep(2000);
    } catch (InterruptedException ex) {
        Logger.getLogger(GameView.class.getName()).log(Level.SEVERE, null, ex);
    }

    parent.finish();//kill activity and return to menu.

}

// Is the player standing next to a bot?
if (levelMap[x][y] == 'P') {
    if (((y-1 > 0) && ((levelMap[x][y-1] == 'B') || (levelMap[x][y-1] == 'F')) ||
        ((x+1 < X_DIMEN) && ((levelMap[x+1][y] == 'B') || (levelMap[x+1][y] == 'F')) ||
        ((y+1 < Y_DIMEN) && ((levelMap[x][y+1] == 'B') || (levelMap[x][y+1] == 'F')) ||
        ((x-1 > 0) && ((levelMap[x-1][y] == 'B') || (levelMap[x-1][y] == 'F')))) {
        botSmash = true;
        smashCoord = new Coord(x,y);
        playerAlmostDead = true;
    }
} else

// Work with rocks and diamonds
if ((levelMap[x][y] == 'O') || (levelMap[x][y] == '+') || (amoebaSmash || botSmash)) {
    if (levelMap[x][y] == 'O') {
        me = 'O';
    } else
    if (levelMap[x][y] == '+') {
        me = '+';
    } else {
        me = '?';
    }
}

// Gravity first...
if ((me != '?') && (levelMap[x][y+1] == '.')) {
    levelMap[x][y+1] = me;
    levelMap[x][y] = '.';
    redraw(new Coord(x,y));
    redraw(new Coord(x,y+1));
    //parent.playSound(R.raw.boing);
    if ((y+2 < Y_DIMEN) && ((levelMap[x][y+2] == 'B') || (levelMap[x][y+2] == 'F') ||
(levelMap[x][y+2] == 'P')))) {
        rockSmash = true;
        smashCoord = new Coord(x,y+2);

```

```

    }
    } else

    // ... then handle stack collapsing
    if ((me != '?') && ((levelMap[x][y+1] == 'O') || (levelMap[x][y+1] == '+'))) {
        if ((levelMap[x+1][y+1] == '.') && (levelMap[x+1][y] == '.')) {
            levelMap[x+1][y+1] = me;
            levelMap[x][y] = '.';
            redraw(new Coord(x,y));
            redraw(new Coord(x+1,y+1));
            if ((y+2 < Y_DIMEN) && ((levelMap[x+1][y+2] == 'B') && (levelMap[x+1][y+2] == 'F') ||
(levelMap[x+1][y+2] == 'P')))) {
                rockSmash = true;
                smashCoord = new Coord(x+1,y+2);
            }
        } else
        if ((levelMap[x-1][y+1] == '.') && (levelMap[x-1][y] == '.')) {
            levelMap[x-1][y+1] = me;
            levelMap[x][y] = '.';
            redraw(new Coord(x,y));
            redraw(new Coord(x-1,y+1));
            if ((y+2 < Y_DIMEN) && ((levelMap[x-1][y+2] == 'B') && (levelMap[x-1][y+2] == 'F') ||
(levelMap[x-1][y+2] == 'P')))) {
                rockSmash = true;
                smashCoord = new Coord(x-1,y+2);
            }
        }
    }
}

// Smash a butterfly, fly, or player
label1: if ((botSmash || amoebaSmash || rockSmash) && ((levelMap[smashCoord.x][smashCoord.y]
== 'B') || (levelMap[smashCoord.x][smashCoord.y] == 'F') || (levelMap[smashCoord.x][smashCoord.y]
== 'P')))) {
    boolean diamond = (levelMap[smashCoord.x][smashCoord.y] == 'B')?true:false;
    if (levelMap[smashCoord.x][smashCoord.y] == 'P') {
        if (botSmash || rockSmash) {
            playerAlmostDead = true;
            parent.playSound(R.raw.explosion_sound);
            Vibrator v = (Vibrator) parent.getSystemService(Context.VIBRATOR_SERVICE);
            v.vibrate(300);
        }
    } else break label1;
}
for (int i=smashCoord.x-1;i<=smashCoord.x+1;i++) {
    for (int u=smashCoord.y-1;u<=smashCoord.y+1;u++) {
        if ((levelMap[i][u] == '.') || (levelMap[i][u] == ':') || (levelMap[i][u] == 'W') ||
(levelMap[i][u] == 'M') || (levelMap[i][u] == '+') || (levelMap[i][u] == 'A')
|| ((i==smashCoord.x) && (u==smashCoord.y-1)) || ((i==smashCoord.x) &&
(u==smashCoord.y)) && (i > 0) && (i < X_DIMEN) && (u > 0) && (u < Y_DIMEN)) {
            if (diamond) {
                levelMap[i][u]='+';
            } else {
                levelMap[i][u]='*';
            }
            redraw(new Coord(i,u));
        }
        for (int b=0; b<bots.size(); b++) {
            BotPosition bot = (BotPosition)bots.get(b);
            if ((bot.x == smashCoord.x) && (bot.y == smashCoord.y)) {
                bots.remove(b);
            }
        }
    }
}

```

```

    }
  }
}

// Go into a magic wall
if ((levelMap[x][y] == 'M') && (y > 1) && ((levelMap[x][y-1] == 'O') || (levelMap[x][y-1] ==
'+')))) {
  if (levelMap[x][y+1] == '.') {
    if (levelMap[x][y-1] == 'O') {
      levelMap[x][y+1] = '+';
      redraw(new Coord(x,y+1));
    } else {
      levelMap[x][y+1] = 'O';
      redraw(new Coord(x,y+1));
    }
  }
  levelMap[x][y-1] = '.';
  redraw(new Coord(x,y-1));
}

// Grow the amoeba
if (levelMap[x][y] == 'A') {
  if (timer >= maxtimer) {
    if (Math.random() < 0.10) {
      int dx=0,dy=0;
      if (Math.random() < 0.5) {
        dx = (int)((Math.round(Math.random()*2))-1);
        dy = 0;
      } else {
        dy = (int)((Math.round(Math.random()*2))-1);
        dx = 0;
      }
      // Spread it only to places that are empty or have dirt
      if ((x+dx < X_DIMEN) && (x+dx > 0) && (y+dy < Y_DIMEN) && (y+dy > 0)) {
        if ((levelMap[x+dx][y+dy] == '.') || (levelMap[x+dx][y+dy] == ':')) {
          levelMap[x+dx][y+dy] = 'A';
          redraw(new Coord(x+dx,y+dy));
        }
      }
    }
  }
}
} // for
} // for

// Move any Fireflies clockwise, and Butterflies counterclockwise
if (timer >= maxtimer) {
  for (int b = 0; b < bots.size(); b++) {
    BotPosition bot = (BotPosition)bots.get(b);
    Coord c = new Coord(bot.x,bot.y);

    if (bot.heading == UP) {
      if ((bot.direction == UP) && (levelMap[bot.x+1][bot.y] == '.')) {
        bot.x++;
        bot.heading = RIGHT;
      } else
        if ((bot.direction == RIGHT) && (levelMap[bot.x-1][bot.y] == '.')) {

```

```

        bot.x--;
        bot.heading = LEFT;
    } else
    if (levelMap[bot.x][bot.y-1] == '.') {
        bot.y--;
    } else {
        bot.heading = DOWN;
    }
}

if (bot.heading == RIGHT) {
    if ((bot.direction == UP) && (levelMap[bot.x][bot.y+1] == '.')) {
        bot.y++;
        bot.heading = DOWN;
    } else
    if ((bot.direction == RIGHT) && (levelMap[bot.x][bot.y-1] == '.')) {
        bot.y--;
        bot.heading = UP;
    } else
    if (levelMap[bot.x+1][bot.y] == '.') {
        bot.x++;
    } else {
        bot.heading = LEFT;
    }
}

if (bot.heading == DOWN) {
    if ((bot.direction == UP) && (levelMap[bot.x-1][bot.y] == '.')) {
        bot.x--;
        bot.heading = LEFT;
    } else
    if ((bot.direction == RIGHT) && (levelMap[bot.x+1][bot.y] == '.')) {
        bot.x++;
        bot.heading = RIGHT;
    } else
    if (levelMap[bot.x][bot.y+1] == '.') {
        bot.y++;
    } else {
        bot.heading = UP;
    }
}

if (bot.heading == LEFT) {
    if ((bot.direction == UP) && (levelMap[bot.x][bot.y-1] == '.')) {
        bot.y--;
        bot.heading = UP;
    } else
    if ((bot.direction == RIGHT) && (levelMap[bot.x][bot.y+1] == '.')) {
        bot.y++;
        bot.heading = DOWN;
    } else
    if (levelMap[bot.x-1][bot.y] == '.') {
        bot.x--;
    } else {
        bot.heading = RIGHT;
    }
}

bots.set(b,bot);

```

```

        if ((bot.x != c.x) || (bot.y != c.y)) {
            levelMap[c.x][c.y] = '.';
            levelMap[bot.x][bot.y] = (bot.direction==0)?'F':'B';
            redraw(new Coord(c.x,c.y));
            redraw(new Coord(bot.x,bot.y));

        }

    }

} // timer

// Check if amoeba is suffocated
boolean amoebaSuffocated = true;
for (int i=1; i<X_DIMEN-1 && amoebaSuffocated; i++) {
    for (int u=1; u<Y_DIMEN-1 && amoebaSuffocated; u++) {
        if (levelMap[i][u] == 'A') {
            if ((levelMap[i][u-1] == '.') || (levelMap[i][u-1] == ':') ||
                (levelMap[i-1][u] == '.') || (levelMap[i-1][u] == ':') ||
                (levelMap[i][u+1] == '.') || (levelMap[i][u+1] == ':') ||
                (levelMap[i+1][u] == '.') || (levelMap[i+1][u] == ':')) {
                amoebaSuffocated = false;
            }
        }
    }
}

// If the amoeba is suffocated, convert all amoeba into diamonds
if (amoebaSuffocated) {
    for (int i=1; i<X_DIMEN-1; i++) {
        for (int u=1; u<Y_DIMEN-1; u++) {
            if (levelMap[i][u] == 'A') {
                levelMap[i][u] = '+';
                redraw(new Coord(i,u));
            }
        }
    }
}

redrawCoords();

if (timer >= maxtimer) {
    timer = 0;
}
timer++;
try {
    Thread.sleep(150);
} catch (InterruptedException e) {
    Log.e("RUN()", e.getMessage());
}
} // while
} // run

}

}

```

```

class Coord {
    int x,y;

    public Coord() {
        x = 0;
        y = 0;
    }

    public Coord(int x, int y) {
        this.x = x;
        this.y = y;
    }
}

class BotPosition {
    int x,y;
    int heading,direction;

    public BotPosition() {
        x = 0;
        y = 0;
        heading = 1;
        direction = 0;
    }

    public BotPosition(int x, int y, int heading, int direction) {
        this.x = x;
        this.y = y;
        this.heading = heading;
        this.direction = direction;
    }
}

```

### **HelpActivity.java**

```

package org.games.boulderdash;

import android.app.Activity;
import android.os.Bundle;
import android.view.ViewGroup.LayoutParams;
import android.webkit.WebView;
import android.widget.LinearLayout;

/**
 *
 * @author AH
 */
public class HelpActivity extends Activity {

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle icle) {
        super.onCreate(icle);
        WebView help =new WebView(this);
        //læser filen lokalt fra telefon
        help.loadUrl("file:///android_asset/BOULDERDASH_HELP.htm");
    }
}

```



```

        LinearLayout help_layout= new LinearLayout(this);
        help_layout.addView(help,LayoutParams.FILL_PARENT,LayoutParams.FILL_PARENT);
        this.setTitle("Boulder Dash Help");
        setContentView(help_layout);
    }
}

```

### **ControlButton.java**

```

package org.games.boulderdash.util;

import org.games.boulderdash.R;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.util.AttributeSet;
import android.util.Log;
import android.widget.Button;

/**
 *
 * @author AH
 */
public class controlButton extends Button{

    public controlButton(Context context, AttributeSet as ){
        super(context, as);
        this.setBackgroundColor(Color.TRANSPARENT);

    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        int h=this.getHeight();
        int w=this.getWidth();
        Paint paint= new Paint();
        paint.setAntiAlias(true);
        paint.setStyle(Paint.Style.FILL);
        paint.setStrokeWidth(2);
        paint.setColor(Color.RED);
        paint.setAlpha(150);
        Path path = new Path();

        int id =this.getId();

        switch(id){

        case R.id.up:
            path.moveTo(w/2, 0);
            path.lineTo(w, h);
            path.lineTo(0, h);
            break;
        case R.id.right:

```

```

        path.moveTo(0, 0);
        path.lineTo(w, h/2);
        path.lineTo(0, h);
        break;
    case R.id.left:
        path.moveTo(0, h/2);
        path.lineTo(w, h);
        path.lineTo(w, 0);
        break;

    case R.id.down:
        path.moveTo(0, 0);
        path.lineTo(w/2, h);
        path.lineTo(w, 0);
        break;
    default:
        path.moveTo(0, 0);
        path.lineTo(w, 0);
        path.lineTo(w, h);
        path.lineTo(0, h);
        paint.setColor(Color.BLACK);
        paint.setAlpha(0);
    }
    path.close();
    canvas.drawPath(path, paint);
}
}

```

## game.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@color/background"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"

    >
    <org.games.boulderdash.GameView
        android:id="@+id/gameView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
    >
</org.games.boulderdash.GameView>
<LinearLayout
    android:orientation="vertical"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    >
    <TextView android:id="@+id/score_board"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:background="@color/text_background"
        android:layout_gravity="center" />

    </LinearLayout>
<TableLayout

```

```

        android:gravity="bottom"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:stretchColumns="*"
    >
    <TableRow
        android:gravity="center"
    >
        <org.games.boulderdash.util.controlButton
            android:id="@+id/up">
        </org.games.boulderdash.util.controlButton>

    </TableRow>
    <TableRow
        android:gravity="center"
    >
        <org.games.boulderdash.util.controlButton
            android:id="@+id/left"
        >
        </org.games.boulderdash.util.controlButton>
        <org.games.boulderdash.util.controlButton
            android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:layout_gravity="center"
        >
        </org.games.boulderdash.util.controlButton>
        <org.games.boulderdash.util.controlButton
            android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:layout_gravity="center"
        >
        </org.games.boulderdash.util.controlButton>
        <org.games.boulderdash.util.controlButton
            android:id="@+id/right"
        >
        </org.games.boulderdash.util.controlButton>
    </TableRow>
    <TableRow
        android:gravity="center"
    >
        <org.games.boulderdash.util.controlButton
            android:id="@+id/down"
        >
        </org.games.boulderdash.util.controlButton>
    </TableRow>
</TableLayout>

</FrameLayout>

```

## main.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@color/background"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"

```

```

android:padding="15dip"
android:orientation="horizontal">
<LinearLayout
    android:orientation="vertical"
    android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    android:layout_gravity="center"
    android:paddingLeft="20dip"
    android:paddingRight="20dip">
    <TextView
        android:text="@string/main_title"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_gravity="center"
        android:layout_marginBottom="20dip"
        android:textSize="24.5sp" />
    <TableLayout
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_gravity="center"
        android:stretchColumns="*">
        <TableRow>
            <Button
                android:id="@+id/new_game"
                android:text="@string/new_game_label" />
        </TableRow>
        <TableRow>
            <Button
                android:id="@+id/highscore_button"
                android:text="@string/highscore_label" />
        </TableRow>
        <TableRow>
            <Button
                android:id="@+id/help_button"
                android:text="@string/help_label" />
        </TableRow>
    </TableLayout>
</LinearLayout>
</LinearLayout>

```

## strings.xml

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Boulder Dash</string>
    <string name="main_title">Android Boulder Dash</string>
    <string name="highscore_label">High Score</string>
    <string name="new_game_label">New Game</string>
    <string name="help_label">Help</string>
    <string name="exit_label">Exit</string>

    <string name="help_title">About Android Boulder Dash</string>

    <string name="levels">10</string>
    <string name="LEVEL_TITLE_1">The Caverns Await</string>
    <string name="LEVEL_AUTHOR_1">Dave Koelle</string>

```

```
<string-array name="NEEDED_DIAMONDS">  
    <item>10</item>  
    <item>36</item>  
</string-array>  
  
<string-array name="POINTS_PER_DIAMOND">  
    <item>100</item>  
    <item>50</item>  
</string-array>  
  
<string-array name="POINTS_PER_EXTRA_DIAMOND">  
    <item>150</item>  
    <item>200</item>  
</string-array>  
  
<string name="IMAGEFILE_1">images.gif</string>  
  
<string-array name="level_data_1">  
    <item>\#####\</item>  
    <item>\#P::WWW:OO:#OOO+#\</item>  
    <item>\#:::O.+WO+:#O++OO#\</item>  
    <item>\#::WW::::W:###:##\</item>  
    <item>\#:...W:O::::::::#\</item>  
    <item>\#:O.O:W::O:.....#\</item>  
    <item>\#WWWWW::W:.....#\</item>  
    <item>\#+O+#####O:#####\</item>  
    <item>\#O+#####WWWWW#\</item>  
    <item>\#O#####W:####\</item>  
    <item>\#:::.....F:OW:####\</item>  
    <item>\#.....W:####\</item>  
    <item>\#.....W:####\</item>  
    <item>\#:::++:::WWWWW#\</item>  
    <item>\#:::O:++:::####\</item>  
    <item>\#:B.....O:++:O#\</item>  
    <item>\#.....O:++:#\</item>  
    <item>\#.....#\\</item>  
    <item>\#:::~::~:]#\</item>  
    <item>\#####\</item>  
</string-array>  
  
<string-array name="level_data_2">  
    <item>\#####\</item>  
    <item>\#:O:::OOO:::O:####\</item>  
    <item>\#:O:::#####O:####\</item>  
    <item>\#:::~::~:]#\</item>  
    <item>\#WWWW:::~::~:]#\</item>  
    <item>\#:::~::~:]#\</item>  
    <item>\#:::~::~:]#\</item>  
    <item>\#:::~::~:]#\</item>  
    <item>\#B.....B.....#\</item>  
    <item>\#..#####.B#\</item>  
    <item>\#..#O::O::O::O:####\</item>  
    <item>\#..#:::~::~:]#\</item>  
    <item>\#..#:::~::~:]#\</item>  
    <item>\#..#.....#..#\</item>  
    <item>\#..#.....P.....#..#\</item>  
    <item>\#..#.....#..#\</item>  
    <item>\#.....#\\</item>  
    <item>\#..#####.B#\</item>
```

```

        <item>\#B.....B.....]#\</item>
        <item>\#####\</item>
    </string-array>
</resources>

```

## AndroidManifest.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.games.boulderdash">
    <uses-permission android:name="android.permission.VIBRATE"/>
    <application android:debuggable="true"
        android:icon="@drawable/boulder_dash">
        <activity android:name="org.games.boulderdash.BoulderDash"
            android:label="BoulderDash" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity android:name="org.games.boulderdash.GameActivity"
            android:screenOrientation="portrait">
        </activity>
        <activity android:name="org.games.boulderdash.HelpActivity">
        </activity>
        <activity android:name="org.games.boulderdash.HighScoreActivity">
        </activity>
    </application>
</manifest>


```








## Boulderdash\_help.html

### BOULDERDASH HELP

Indsaml så mange diamanter som muligt

### BOULDERDASH HJÆLP

Indsaml så mange diamanter  som muligt, men undgå faldende klipper og sværmende ildfluer. Undervejs vil du støde på disse elementer.

<b>Titanium Wall</b>		Mur der ikke kan ødelægges
<b>Dirt</b>		Du kan grave dig gennem jord, og det efterlader tom plads.
<b>Brick Wall</b>		Denne type mur kan ødelægges hvis du sprænger en ildflue i luften ved siden af den.
<b>Magic Wall</b>		Kan også ødelægges. Hvis du taber en sten på denne type mur bliver den forvandlet til en diaman. Hvis du taber en diaman bliver den til sten. Der skal være luft under muren ellers bliver de faldende elementer opslugt af muren.
<b>Firefly</b>		Bevæger sig med uret. Eksploderer hvis du rammer den med en sten eller en diaman. Dødbringende at røre.
<b>Butterfly</b>		Bevæger sig mod uret. Bliver til Diamanter hvis du rammer den med en sten. Dødbringende at røre.
<b>Closed Exit</b>		Når du har samlet det nødvendige antal diamanter forvandles den til en dør til næste level.
<b>Opened Exit</b>		Passage til næste level.