כפר הנוער "עתיד" ע"ש יוענה ז'בוטינסקי

Car Hell

מגיש: אריאל גוב

ת.ז: 212642367

מורה: לאוניד שפרן

חלופה: תכנון ותכנות מערכות טלפונים חכמים

בית ספר: יוהנה ז'בוטינסקי

תאריך הגשה: 30/5/20

תוכן עניינים

תוכן

מבוא

בתיק פרויקט הזה אני אסביר על הפרויקט שלי "Car Hell". Car Hell הינו משחק לטלפון ובו אתה משחק בתור מכונית. מטרת המשחק היא לסרות כמה שיותר זמן תוך כדי שאתה מנסה להתחמק מ"יריות" אשר עפות ברחבי המסך ומנסות לפגוע בך.

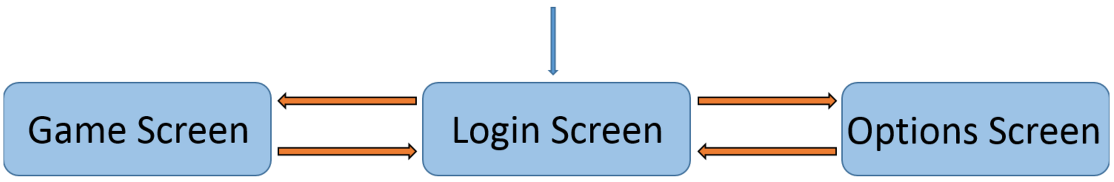
הרעיון לפרויקט הגיע כאשר המורה אמר לנו שאלינו לבחור נושא לעבודת הגמר. הרעיון המקורי היה ליצור משחק בסגנון "bullet hell". משום מה רציתי גם לעשות משחק עם מכונית, וכך עלה הרעיון לחבר בין המשחקים וליצור את המשחק הזה.

תיאור התוכנה

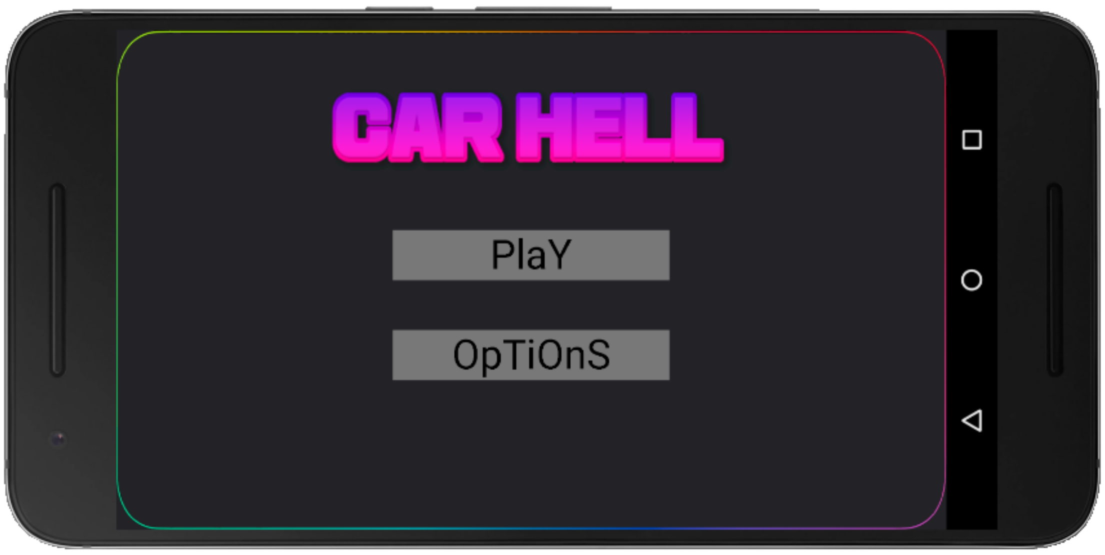
התכנה הינה משחק פשוט בסגנון "bullet hell", אבל השחקן הוא מכונית.

המטרה במשחק היא לשרוד כמה שיותר זמן ולהתחמק מיריות שעפות ברחבי המסך במהירויות ובכיוונים שונים.

למשחק ישנם 3 מסכים שונים, login, game ו- options.

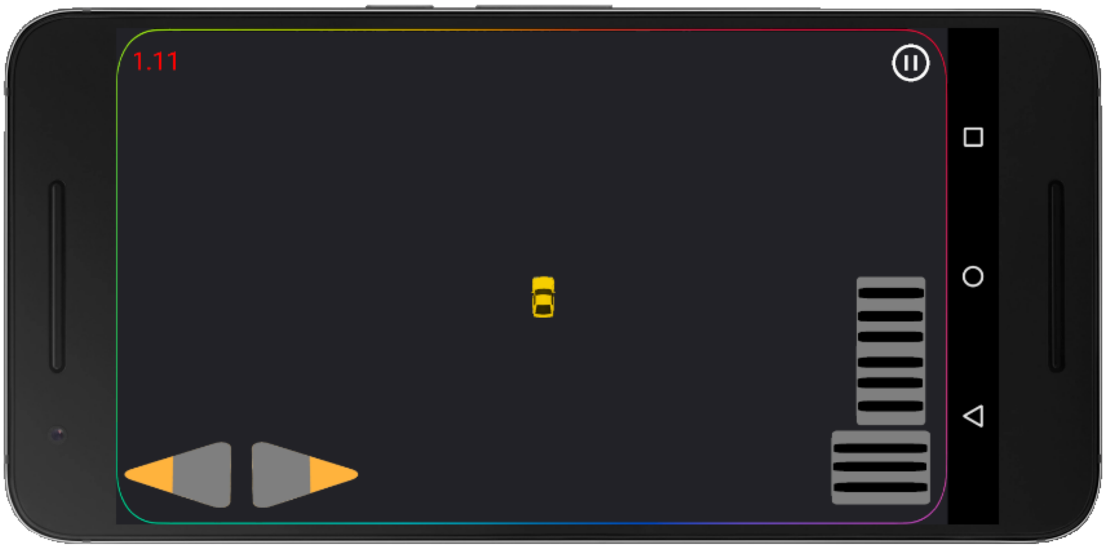


Login Screen



כשהמשתמש פותח את האפליקציה הוא נמצא ב-login screen, מסך הכניסה. זהוא בעצם המסך הכי פשוט והשימוש העיקרי בו הוא על מנת לנווט בין המסכים האחרים, במסך זה יש את שם התכנה ושני כפתורים: PlaY ו- OpTiOnS.

Game Screen



לחיצה על כפתור ה"PlaY" תעביר את המשתמש למסך המשחק, Game Screen, שם בעצם מתנהל המשחק. במסך המשחק ישנו השחקן, כפתור גז, ברקס, ימינה ושמאלה.

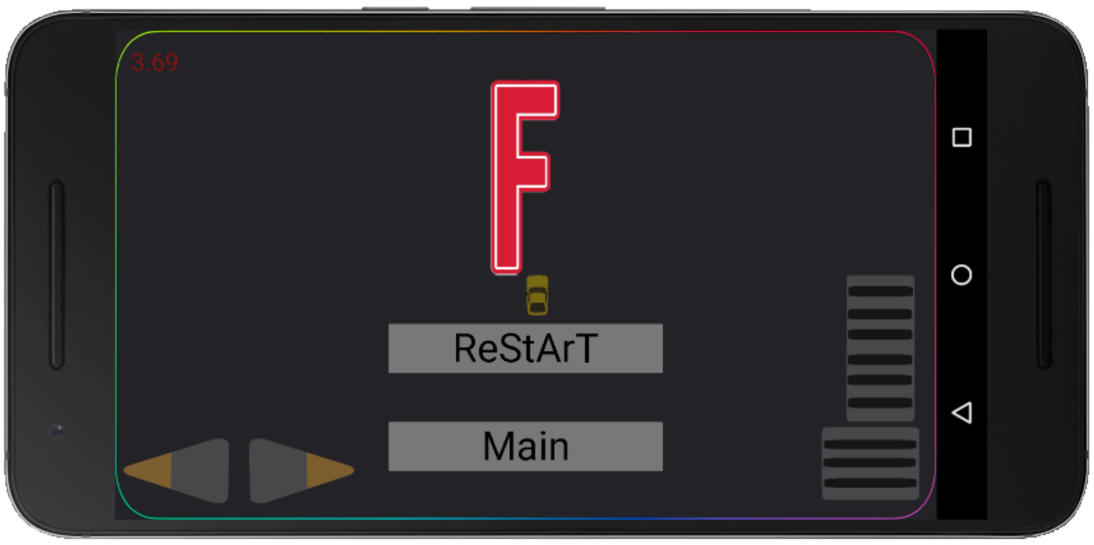
לחיצה על כפתור הגז תגרום למכונית (שחקן) להאיץ קדימה עד למהירות מסויימת, השחקן לא מתחיל לזוז במהירות מקסימלית מייד אלה צריך להאיץ אליה. וכאשר השחקן עוזב את כפתור הגז המכונית תאט עד שהיא תעצור. כפתור הברקס מזרז את המהירות שבה המכונית עוצרת ומשמש לנסיעה ברוורס. הכפתורים של ימינה ושמאלה מסובבים את המכונית, אבל אפשר לפנות רק אם המכונית נוסעת. אם היא עומדת במקום היא לא פונה.

גבולות המסך הם גם גבולות המשחק, אם השחקן נוסעה לתוך אחד הצדדים אז הוא יופיע מהצד השני, כמו בפאקמן-מן. זה כולל את ארבעת הצדדים, למעלה, למטה, ימינה, שמאלה.

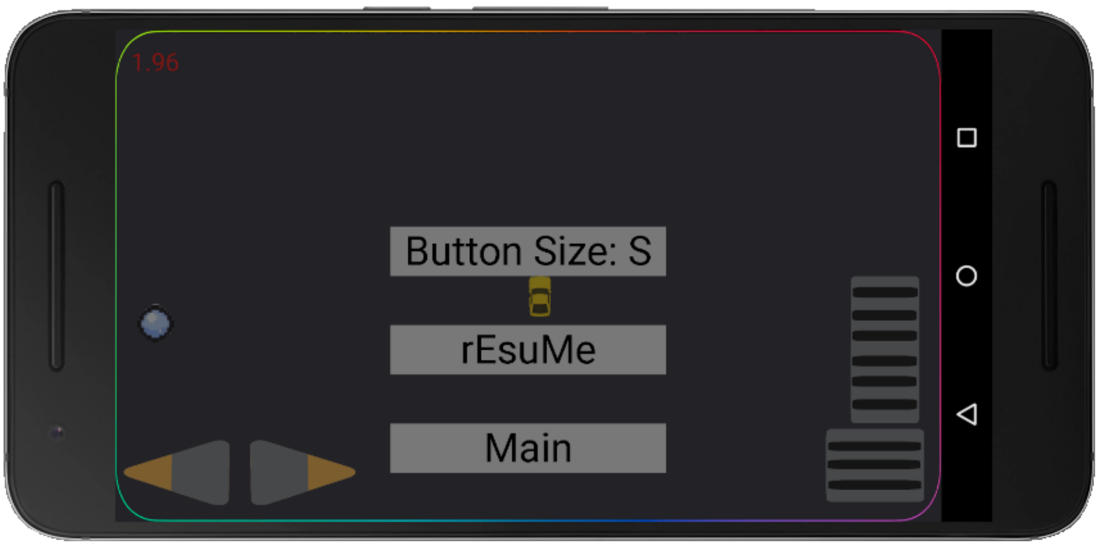
כרגע במשחק ישנם שלוש סוגים שונים של יריות שעפות ברחבי המסך. כשהמשחק מתחיל רק יריות כחולות שעפות לכיוונים רנדומליים יווצרו. לאחר 10 שניות יתחילו להופיע גם יריות אדומות/ ורודות, שהן היריות הכי איטיות, אבל הן עפות לכיוון שבו השחקן היה כשהן נוצרו. אם השחקן שורד יותר מ-25 שניות אז יריות ירוקות יתחילו להופיע, היריות הירוקות הן ההכי מהירות אבל הן עפות רק בקווים ישרים.

על מנת לעזור לשחקן לשרוד פעם בכמה זמן מופיע על המסך מגן, המגן נשאר לכמות זמן מוגבלת והוא מהבהב לפני שהוא נעלם. אם השחקן נוגע במגן אז במשך 5 שניות השחקן הופך למוגן מפני פגיעה של היריות.

במידה והשחקן נפגע מאחת היריות המסך הבא יופיע:



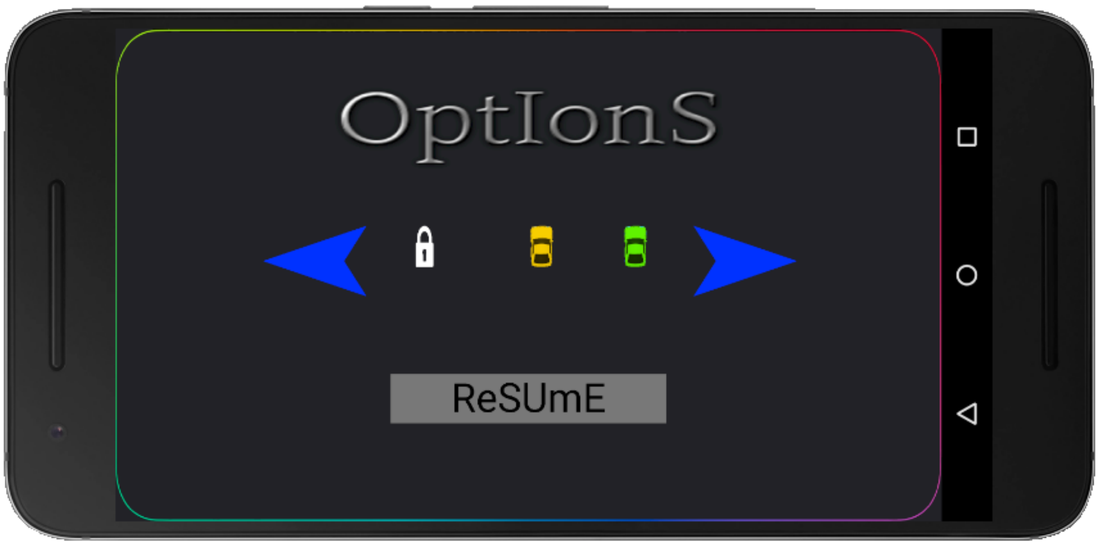
בכל רגע במהלך המשחק השחקן יכול ללחוץ על כפתור ה-"pause" שנמצא בצד ימין למעלה, לחיצה על הכפתור תעצור את המשחק, תעמעם את הרקע ותפתח שלוש כפתורים חדשים.



כפתור "rEsuMe" שממשיך את המשחק מהנקודה שעצרתה בה, כפתור "Main" שלחיצה עליו מחזירה את המשתמש למסך הראשי (מסך הכניסה), וכפתור נוסף "Button Size :S" או "Button Size :L". כשהמשתמש מפעיל את התכנה בפעם הראשונה הכפתורים נמצאים על מצב קטן וכשהוא יעצור את המשחק יופיע לו הכפתור "Button Size :S", לחיצה על הכפתור

תשנה אותו ל"Button Size :L" ותגדיל את כפתורי המשחק לכפתורים גדולים, לחיצה נוספת על הכפתורים תחזיר אותם להיות קטנים. אם המשתמש בחר לשנות את גודל הכפתורים אז התכנה תזכור את הבחירה שלו ופעם הבאה שהוא יכנס הכפתורים יהיו בהתאם למה שהוא קבע פעם קודמת שהוא השתמש בתכנה.

Options Screen



לחיצה על כפתור ה"OpTiOnS " במסך הכניסה תעביר את המשתמש לסוג של מסך התאמה אישית שבו המשתמש יכול לשנות כיצד נראית המכונית במשחק. כרגע ישנם 8 מכוניות שונות במשחק, מתוכם 3 פתוכות מהרגע הראשון, אבל את 5 המכוניות האחרות השחקן צריך לפתוח בכך שהוא ישחק במשחק, לדוגמה מכונית אחד דורשת מהשחקן לשחק לפחות משחק אחד על מנת לפתוח אותה, בעוד שאחרת דורשת ממנו לשרוד 10 שניות בתוך המשחק. כל עוד השחקן לא פתח את המכונית היא תופיע ב"Options Screen" בתור מנעול, ואת אלה שהו כן פתח הוא יוכל לראות. מעל כל מכונית רשום מה השחקן צריך לעשות על מנת לפתוח אותה.

השחקן מנוות בין המכוניות באמצעות 2 חצים, ימינה ושמאלה. כאשר השחקן יחזור למשחק המכונית שלו תהיה המכונית שהוא בחר, והבחירה שלו, בנוסף לאיזה מכוניות הוא פתח ואיזה לא, נשמרת גם לאחר שהמשתמש סוגר את התכנה.

מדריך משתמש

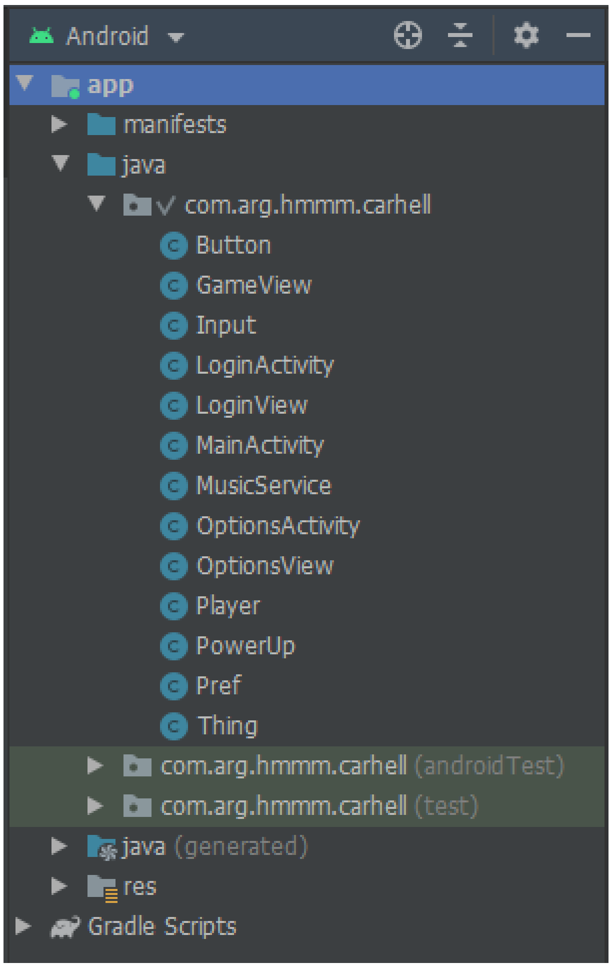
CarHell הוא משחק Bullet hell פשוט יחסית שבו השחקן שולט על מכונית.

כאשר אתה נכנס למשחק יש לך 2 אפשרויות, PlaY ו OpTiOnS. כדי להתחיל לשחק יש ללחוץ על כפתור ה"PlaY". בתוך המשחק המתרה היא לשרוד כמה שיותר זמן, לחיצה על דוושת הגז תגרום למכונית לשוע קדימה ולחיצה על דוושת הברקס תעצור את המכונית והיא תתחיל לנסוע אחורה. לחיצה על הכפתור בצד ימין למעלה תעצור את המשחק ותיתן את האפשרות להמשיך את המשחק או לחזור למסך ההתחלה, חזרה למסך ההתחלה תיחשב בתור הפסד. בנוסף לכך ישנה האפשרות לשנות את הגודל של הכפתורים במשחק מקטן לגדול, הבחירה תישמר גם אחרי שסגרת את המשחק.

לחיצה על כפתור ה"OpTiOnS" תפתח את חלון האפשרויות, בו אפשר לשנות את איך שהשחקן נראה. כדי לעשות זאת יש ללחוץ על החצים הכחולים במרכז המסך, לחיצה על החץ תדפדף בין המכוניות השונות. לא כל המכוניות פתוחות לשימוש על ההתחלה, על מנת לפתוח מכוניות נוספות יש לעמוד בתנאי שרשום מעליהם, לכל מכונית נעולה יש תנאי משלה כדי לפתוח אותה. כל מכונית שאתה פותח תישאר פתוחה גם אחרי שהמשחק נסגר, והבחירה שלך למכונית תישאר גם היא כך שפעם הבאה שאתה נכנס למשחק אתה יכול להתחיל מייד עם אותה מכונית שבחרת פעם קודמת ועל כפתורים גדולים/קטנים בהתאם לבחירתך.

מדריך למפתח

נתונים:



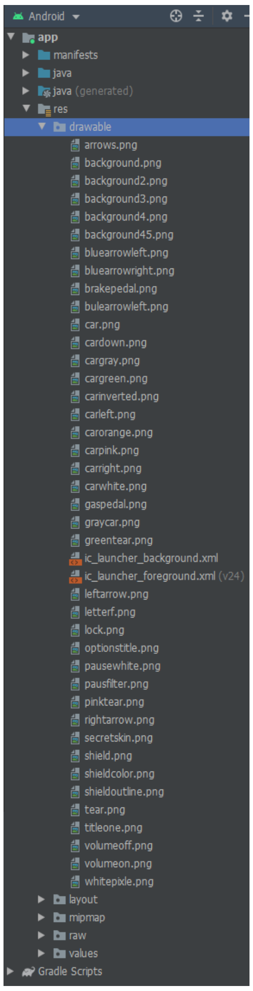
אלה הם כל המחלקות אשר נמצאות בפרויקט.

לכל מסך יש View ו- Activity משלו.

- **MainActivity ו- GameView** הם המחלקות העקריות של מסך המשחק
- **LoginActivity ו- LoginView** הם המחלקות העקריות של מסך הכניסה
- **OptionsView ו- OptionsActivity** הם המחלקות העקריות של מסך האפשרויות
- **Input** משתמש ב- המחלקה OnTouchListener הזו קולטת את המגע של המשתמש במסך
- **Button** מייצג את הכפתורים במשחק והוא מקבל קלט מ**Input** כדי לזהות אם הכפתור נלחץ או לא
- **Thing** מייצג כל עצם שנמצא במשחק (חוץ מכפתורים). **Thing** מקבל (x,y, GameView) או שהוא מקבל (x,y, GameView, vx, vy, Bitmap), vx,vy הם המהירות על ציר הx/y

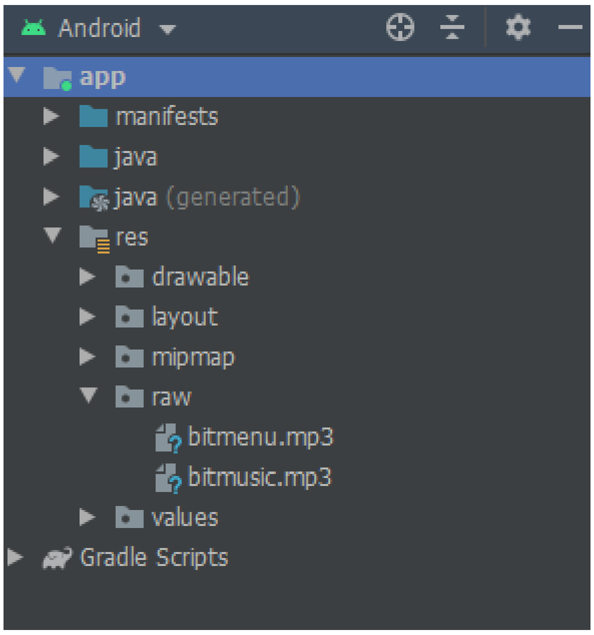
- **Player** מייצג את השחקן והוא יורש ממחלקה **Thing**
- **PowerUp** מייצג את הדברים שעוזרים לשחקן(כרגע זה רק המגן) והוא יורש ממחלקה **Thing**
- **Pref** משמש לשמירת נתונים גם לאחר שהמשתמש סגר את התוכנה
- **MusicService** הוא service שאחראי על המוזיקה במשחק

בתקייה drawable שמורות כל התמונות שנמצאות בשימוש במשחק.



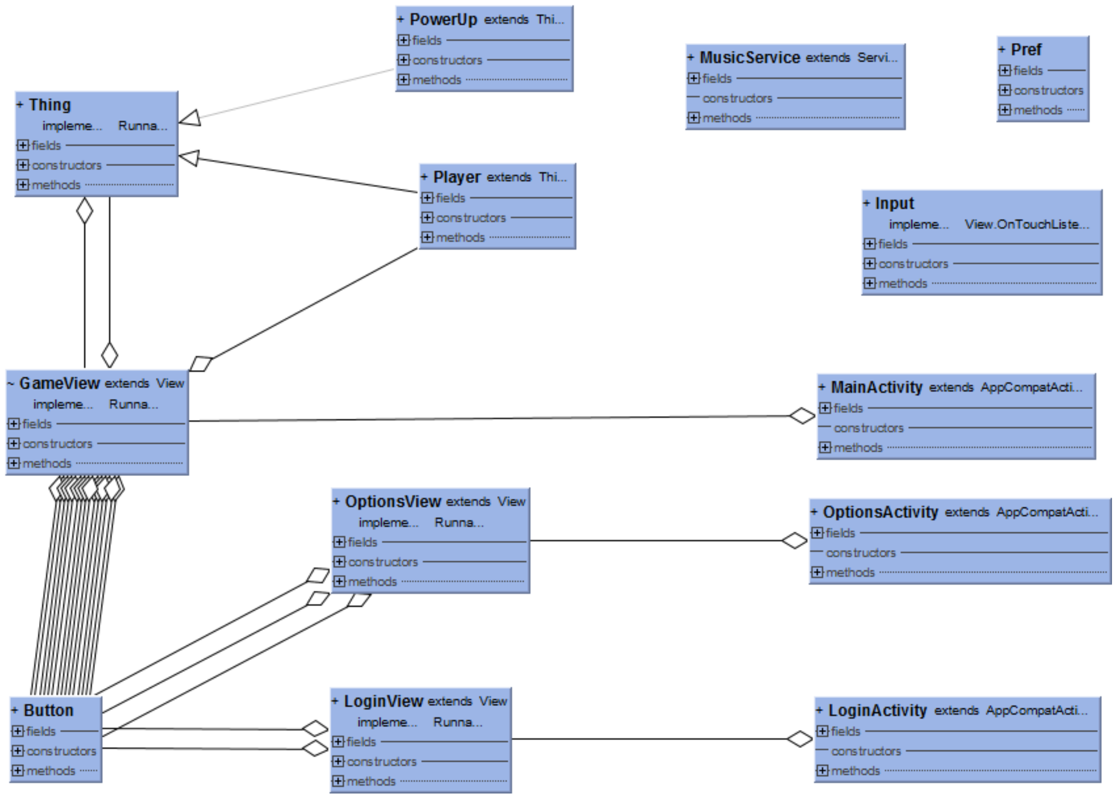
```
Android
▼ app
  ▶ manifests
  ▶ java
  ▶ java (generated)
  ▼ res
    ▼ drawable
        arrows.png
        background.png
        background2.png
        background3.png
        background4.png
        background45.png
        bluearrowleft.png
        bluearrowright.png
        brakepedal.png
        bulearrowleft.png
        car.png
        cardown.png
        cargray.png
        cargreen.png
        carinverted.png
        carleft.png
        carorange.png
        carpink.png
        carright.png
        carwhite.png
        gaspedal.png
        graycar.png
        greentear.png
        ic_launcher_background.xml
        ic_launcher_foreground.xml (v24)
        leftarrow.png
        letterf.png
        lock.png
        optionstitle.png
        pausewhite.png
        pausfilter.png
        pinktear.png
        rightarrow.png
        secretskin.png
        shield.png
        shieldcolor.png
        shieldoutline.png
        tear.png
        titleone.png
        volumeoff.png
        volumeon.png
        whitepixle.png
    ▶ layout
    ▶ mipmap
    ▶ raw
    ▶ values
▶ Gradle Scripts
```

בתקייה raw שמורות כל המנגינות שנמצאות במשחק



```
Android
▼ app
  ▶ manifests
  ▶ java
  ▶ java (generated)
  ▼ res
    ▶ drawable
    ▶ layout
    ▶ mipmap
    ▼ raw
        bitmenu.mp3
        bitmusic.mp3
    ▶ values
▶ Gradle Scripts
```

:UML



Extends - ▽

Assosiation - ◇

:JavaDoc

MainActivity

2020-04-15

com.arg.hmmm.carhell

Class MainActivity

- java.lang.Object
  - com.arg.hmmm.carhell.MainActivity

```
public class MainActivity
extends java.lang.Object
```

–

Field Summary

Fields

Modifier and Type	Field and Description
`private GameView`	`gameView`

–

Method Summary

All Methods *Instance Methods* *Concrete Methods*

Modifier and Type	Method and Description
`protected void`	`onCreate(Bundle savedInstanceState)` creates the main activity
`protected void`	`onResume()` resumes the music when the activity resumes
`protected void`	`onStop()` stops the music when the activity stops

•

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

–

Field Detail

- *gameView*

  `private GameView gameView`

–

Method Detail

- *onCreate*

  `protected void onCreate(Bundle savedInstanceState)`

  creates the main activity

  Parameters:

  `savedInstanceState` -

- *onStop*

  `protected void onStop()`

  stops the music when the activity stops

- *onResume*

  `protected void onResume()`

  resumes the music when the activity resumes

GameView

2020-04-15

com.arg.hmmm.carhell

Class GameView

- java.lang.Object
  - View
    - com.arg.hmmm.carhell.GameView
- **All Implemented Interfaces:**
- java.lang.Runnable

```
class GameView
extends View
implements java.lang.Runnable
```

–

Field Summary

Fields

Modifier and Type	Field and Description
`private Bitmap`	`background`
`private Button`	`breakpedalL`
`private Button`	`breakpedalS`
`private Button`	`buttonSizeChange`
`private java.text.DecimalFormat`	`df2`
`private Bitmap`	`filterbmp`
`private Paint`	`filterpaint`
`private Button`	`gaspedalL`
`private Button`	`gaspedalS`
`private boolean`	`isdead`
`private Button`	`leftarrowL`
`private Button`	`leftarrowS`
`private boolean`	`needToStop`
`private Paint`	`paint`

private Button	pauseButton
private boolean	paused
private Player	player
private Button	restart
private Button	resumeButton
private Button	returnToMainButton
private Button	rightarrowL
private Button	rightarrowS
private Bitmap	ripbmp
private int	screenHeight
private int	screenWidth
private float	seconds
private java.util.TimerTask	secondsTask
private java.util.TimerTask	summonPowerUpTask
private java.util.TimerTask	summonTearTask
private int	summonTearTaskPeriod
private Thing	testPixle used for testing
private java.util.ArrayList<Thing>	things the things array stores all of the bullets/powerups ingame, adding/removing things from it as it goes
private Button	volumeOffButton
private Button	volumeOnButton

–

Constructor Summary

Constructors

Constructor and Description
`GameView(Context context)` starts the game

–

Method Summary

All Methods *[Instance Methods](#)* *[Concrete Methods](#)*

Modifier and Type	Method and Description
`void`	`addThing(Thing thing)` adds thing to the array
`private void`	`createButtons()` generates all of the buttons
`private void`	`createPlayer()` generates the player
`Player`	`getPlayer()` returns the player
`int`	`getScreenHeight()` returns the screen height
`int`	`getScreenWidth()` returns the screen width
`Thing`	`getTestPixle()` returns the test pixle (used for testing)
`java.util.ArrayList<Thing>`	`getThings()` returns things array
`void`	`looseScrean()` swiches to loose screan
`protected void`	`onDraw(Canvas canva)` draws everything in the game
`private void`	`pauseScrean()` switches to pause screen
`void`	`removeThing(Thing thing)` removes thing from the things array
`private void`	`restartView()` restart the run, restarting the game
`void`	`resume()` resumes the thread
`private void`	`returnToLogin()`

	returns to login screen
`void`	`run()` checks if any of the buttons have been pressed
`private void`	`startPowerUpTimer()` start the power up timer power up timer responsible for the power up spawn rate
`private void`	`startSecondsTimer()` start the seconds timer seconds timer counts the seconds from the start of the run
`private void`	`startTearTimer()` start the tear timer tear timer is responsible for the tear spawn rate
`private void`	`stopPause()` stops pauses screen and returns to the game
`void`	`summonTear()` summons a tear

-

  ### Methods inherited from class java.lang.Object

  `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

- *player*

  ```
  private Player player
  ```

- *things*

  ```
  private java.util.ArrayList<Thing> things
  ```

  ```
  the things array stores all of the bullets/powerups
  ingame, adding/removing things from it as it goes
  ```

- *screenWidth*
  `private int screenWidth`

- *screenHeight*
  `private int screenHeight`

- *background*
  `private Bitmap background`

- *seconds*
  `private float seconds`

- *paint*
  `private Paint paint`

- *summonTearTask*
  `private java.util.TimerTask summonTearTask`

- *summonTearTaskPeriod*
  `private int summonTearTaskPeriod`

- *secondsTask*
  `private java.util.TimerTask secondsTask`

- *summonPowerUpTask*
  `private java.util.TimerTask summonPowerUpTask`

- *needToStop*
  `private boolean needToStop`

- *isdead*
  `private boolean isdead`

- *paused*
  `private boolean paused`

- *ripbmp*
  `private Bitmap ripbmp`

- *filterbmp*
  `private Bitmap filterbmp`

- *filterpaint*
  `private Paint filterpaint`

- *df2*
  `private java.text.DecimalFormat df2`

- *testPixle*
  `private Thing testPixle`

  `used for testing`

- *gaspedalS*
  `private Button gaspedalS`

- *breakpedalS*
  `private Button breakpedalS`

- *leftarrowS*
  `private Button leftarrowS`

- *rightarrowS*
  `private Button rightarrowS`

- *gaspedalL*
  `private Button gaspedalL`

- *breakpedalL*
  `private Button breakpedalL`
- *leftarrowL*
  `private Button leftarrowL`
- *rightarrowL*
  `private Button rightarrowL`
- *restart*
  `private Button restart`
- *returnToMainButton*
  `private Button returnToMainButton`
- *resumeButton*
  `private Button resumeButton`
- *pauseButton*
  `private Button pauseButton`
- *buttonSizeChange*
  `private Button buttonSizeChange`
- *volumeOnButton*
  `private Button volumeOnButton`
- *volumeOffButton*
  `private Button volumeOffButton`

–

Constructor Detail

- *GameView*
  `public GameView(Context context)`

  starts the game

Method Detail

- *onDraw*
  `protected void onDraw(Canvas canvas)`

  draws everything in the game
  Parameters:
  `canvas` -

- *removeThing*
  `public void removeThing(Thing thing)`

  removes thing from the array
  Parameters:
  `thing` -

- *addThing*
  `public void addThing(Thing thing)`

  adds thing to the array
  Parameters:
  `thing` -

- *getThings*
  `public java.util.ArrayList<Thing> getThings()`

  returns things array
  Returns:

- *getTestPixle*
  `public Thing getTestPixle()`

  returns the test pixle (used for testing)
  Returns:

- *getPlayer*
  `public Player getPlayer()`

  returns the player

- *getScreenWidth*
  `public int getScreenWidth()`

  returns the screen width

- *getScreenHeight*
  `public int getScreenHeight()`

  returns the screen height
  Returns:

- *createPlayer*
  `private void createPlayer()`

  generates the player

- *createButtons*
  `private void createButtons()`

  generates all of the buttons

- *startTearTimer*
  `private void startTearTimer()`

  start the tear timer tear timer is responsible for the tear spawn rate

- *startPowerUpTimer*
  `private void startPowerUpTimer()`

  start the power up timer power up timer responsible for the power up spawn rate

- *startSecondsTimer*
  `private void startSecondsTimer()`

  start the seconds timer seconds timer counts the seconds from the start of the run

- *summonTear*
  `public void summonTear()`

  summons a tear

- *looseScrean*
  `public void looseScrean()`

  swiches to loose screan

- *pauseScrean*
  `private void pauseScrean()`

  switches to pause screen

- *stopPause*
  `private void stopPause()`

  stops pauses screen and returns to the game

- *run*
  `public void run()`

  checks if any of the buttons have been pressed
  Specified by:
  `run` in interface `java.lang.Runnable`

- *returnToLogin*
  `private void returnToLogin()`

  returns to login screen

- *resume*
  `public void resume()`

  resumes the thread

- *restartView*
  `private void restartView()`

  restart the run, restarting the game

LoginActivity

2020-04-15

com.arg.hmmm.carhell

Class LoginActivity

- java.lang.Object
  - android.app.Activity
    - androidx.core.app.ComponentActivity
      - androidx.fragment.app.FragmentActivity
        - androidx.appcompat.app.AppCompatActivity
          - com.arg.hmmm.carhell.LoginActivity
- **All Implemented Interfaces:**
- androidx.appcompat.app.ActionBarDrawerToggle.DelegateProvider, androidx.appcompat.app.AppCompatCallback, androidx.core.app.ActivityCompat.OnRequestPermissionsResultCallback, androidx.core.app.ActivityCompat.RequestPermissionsRequestCodeValidator, androidx.core.app.TaskStackBuilder.SupportParentable, androidx.core.view.KeyEventDispatcher.Component, androidx.lifecycle.LifecycleOwner, androidx.lifecycle.ViewModelStoreOwner

---

```
public class LoginActivity
extends androidx.appcompat.app.AppCompatActivity
```

–

Nested Class Summary

-

Nested classes/interfaces inherited from class androidx.core.app.ComponentActivity

`androidx.core.app.ComponentActivity.ExtraData`

–

Field Summary

Fields

Modifier and Type	Field and Description
`private LoginView`	`loginView`
`private Intent`	`musicServiceIntent`

–

Constructor Summary

Constructors

Constructor and Description
`LoginActivity()`

–

Method Summary

All Methods *[Instance Methods](#)* *[Concrete Methods](#)*

Modifier and Type	Method and Description
`protected void`	`onCreate(Bundle savedInstanceState)` creates the login activity login activity is the first screen of the app
`protected void`	`onResume()` resumes the music when the activity resumes
`protected void`	`onStop()` stops the music when the activity stops

- ### Methods inherited from class androidx.appcompat.app.AppCompatActivity

  `addContentView, closeOptionsMenu, dispatchKeyEvent, findViewById, getDelegate, getDrawerToggleDelegate, getMenuInflater, getResources, getSupportActionBar, getSupportParentActivityIntent, invalidateOptionsMenu, onConfigurationChanged, onContentChanged, onCreateSupportNavigateUpTaskStack, onDestroy, onKeyDown, onMenuItemSelected, onMenuOpened, onPanelClosed, onPostCreate, onPostResume, onPrepareSupportNavigateUpTaskStack, onSaveInstanceState, onStart, onSupportActionModeFinished, onSupportActionModeStarted, onSupportContentChanged, onSupportNavigateUp, onTitleChanged, onWindowStartingSupportActionMode, openOptionsMenu, setContentView, setContentView, setContentView, setSupportActionBar, setSupportProgress, setSupportProgressBarIndeterminate, setSupportProgressBarIndeterminateVisibility, setSupportProgressBarVisibility, setTheme, startSupportActionMode, supportInvalidateOptionsMenu, supportNavigateUpTo, supportRequestWindowFeature, supportShouldUpRecreateTask`

-

  Methods inherited from class androidx.fragment.app.FragmentActivity
  `dump, getLastCustomNonConfigurationInstance, getLifecycle, getSupportFragmentManager, getSupportLoaderManager, getViewModelStore, onActivityResult, onAttachFragment, onBackPressed, onCreatePanelMenu, onCreateView, onCreateView, onLowMemory, onMultiWindowModeChanged, onNewIntent, onPause, onPictureInPictureModeChanged, onPrepareOptionsPanel, onPreparePanel, onRequestPermissionsResult, onResumeFragments, onRetainCustomNonConfigurationInstance, onRetainNonConfigurationInstance, onStateNotSaved, setEnterSharedElementCallback, setExitSharedElementCallback, startActivityForResult, startActivityForResult, startActivityFromFragment, startActivityFromFragment, startIntentSenderForResult, startIntentSenderForResult, startIntentSenderFromFragment, supportFinishAfterTransition, supportPostponeEnterTransition, supportStartPostponedEnterTransition, validateRequestPermissionsRequestCode`

-

  Methods inherited from class androidx.core.app.ComponentActivity
  `dispatchKeyShortcutEvent, getExtraData, putExtraData, superDispatchKeyEvent`

-

  Methods inherited from class java.lang.Object
  `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

–

Field Detail

- *loginView*
  `private LoginView loginView`

- *musicServiceIntent*
  `private Intent musicServiceIntent`

–

Constructor Detail

- *LoginActivity*
  `public LoginActivity()`

–

Method Detail

- *onCreate*
  `protected void onCreate(Bundle savedInstanceState)`

  creates the login activity login activity is the first screen of the app
  Overrides:
  `onCreate` in class `androidx.appcompat.app.AppCompatActivity`
  Parameters:
  `savedInstanceState` -

- *onStop*
  `protected void onStop()`

  stops the music when the activity stops
  Overrides:
  `onStop` in class `androidx.appcompat.app.AppCompatActivity`

- *onResume*
  `protected void onResume()`

  resumes the music when the activity resumes
  Overrides:
  `onResume` in class `androidx.fragment.app.FragmentActivity`

LoginView

2020-04-15

com.arg.hmmm.carhell

Class LoginView

- java.lang.Object
  - View
    - com.arg.hmmm.carhell.LoginView
- **All Implemented Interfaces:**
- java.lang.Runnable

```
public class LoginView
extends View
implements java.lang.Runnable
```

–

Field Summary

Fields

Modifier and Type	Field and Description
`private Bitmap`	`background`
`private boolean`	`needToStop2`
`private Button`	`optionsButton`
`private Button`	`playButton`
`(package private) int`	`screenHeight`
`(package private) int`	`screenWidth`
`private Bitmap`	`title`

–

Constructor Summary

Constructors

Constructor and Description
`LoginView(Context context)` creates a login view, in the login the user choses if to start the game or to go to options

–

Method Summary

All Methods *Instance Methods* *Concrete Methods*

Modifier and Type	Method and Description
`private boolean`	`needToStop()` checks if the view needs to stop or not, used in run()
`protected void`	`onDraw(Canvas canvas)` draws everything in the view
`private void`	`openGameActivity()` swiches to the game activity
`private void`	`openOptionsActivity()` swiches to the options activity
`void`	`resume()` resumes the view
`void`	`run()` checks if any of the buttons are pressed

- #### Methods inherited from class java.lang.Object
  `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

–

Field Detail

- *playButton*
  `private Button playButton`

- *optionsButton*
  `private Button optionsButton`

- *needToStop2*
  `private boolean needToStop2`

- *title*
  `private Bitmap title`

- *background*
  `private Bitmap background`

- *screenWidth*
  `int screenWidth`

- *screenHeight*
  `int screenHeight`

–

Constructor Detail

- *LoginView*
  `public LoginView(Context context)`

  creates a login view, in the login the user choses if to start the game or to go to options

  Parameters:

  `context` -

–

Method Detail

- *onDraw*
  `protected void onDraw(Canvas canvas)`

  draws everything in the view

  Parameters:

  `canvas` -

- *run*
  `public void run()`

  checks if any of the buttons are pressed

  Specified by:

  `run` in interface `java.lang.Runnable`

- *openGameActivity*
  `private void openGameActivity()`

  swiches to the game activity

- *openOptionsActivity*
  `private void openOptionsActivity()`

  swiches to the options activity

- *needToStop*
  `private boolean needToStop()`

  checks if the view needs to stop or not, used in run()
  Returns:

- *resume*
  `public void resume()`

  resumes the view

OptionsActivity

2020-04-15

com.arg.hmmm.carhell

Class OptionsActivity

- java.lang.Object
  - com.arg.hmmm.carhell.OptionsActivity

```
public class OptionsActivity
extends java.lang.Object
```

–

Field Summary

Fields

Modifier and Type	Field and Description
`private OptionsView`	`optionsView`

–

Method Summary

All Methods *Instance Methods* *Concrete Methods*

Modifier and Type	Method and Description
`protected void`	`onCreate(Bundle savedInstanceState)` start the options activity
`protected void`	`onResume()` resumes the music when the activity resumes
`protected void`	`onStop()` stops the music when the activity stops

-

Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify,
notifyAll, toString, wait, wait, wait
```

–

Field Detail

- *optionsView*
  `private OptionsView optionsView`

–

Method Detail

- *onCreate*
  `protected void onCreate(Bundle savedInstanceState)`

  start the options activity

  Parameters:

  `savedInstanceState` -

- *onStop*
  `protected void onStop()`

  stops the music when the activity stops

- *onResume*
  `protected void onResume()`

  resumes the music when the activity resumes

OptionsView

2020-04-15

com.arg.hmmm.carhell

Class OptionsView

- java.lang.Object
    - View
        - com.arg.hmmm.carhell.OptionsView
- **All Implemented Interfaces:**
- java.lang.Runnable

```
public class OptionsView
extends View
implements java.lang.Runnable
```

–

Field Summary

Fields

Modifier and Type	Field and Description
`private Bitmap`	`background`
`private Button`	`carScrollLeft`
`private Button`	`carScrollRight`
`private boolean`	`needToStop`
`private int`	`pid` player image pointer used to identify wich car is corrently chosen by the player
`private int`	`pidm` player image pointer minus points to the previous car
`private int`	`pidp` player image pointer plus points to the next car
`private Bitmap[]`	`playerBmp`
`private java.lang.String[]`	`playerImages`

`private Button`	`resumeButton`
`(package private) int`	`screenHeight`
`(package private) int`	`screenWidth`
`private Bitmap`	`title`
`private boolean[]`	`unlocked`
`private Paint`	`unlockPaint`
`private java.lang.String[]`	`unlocktext`
`private float`	`xPosLeft`
`private float`	`xPosMiddle`
`private float`	`xPosRight`

–

Constructor Summary

Constructors

Constructor and Description
`OptionsView(Context context)` creates the options view

–

Method Summary

All Methods *Instance Methods* *Concrete Methods*

Modifier and Type	Method and Description
`protected void`	`onDraw(Canvas canvas)` draws the options view
`private void`	`openLoginActivity()` swiches to the login activity
`void`	`resume()` resumes corrent activity
`void`	`run()` checks what buttons are being pressed
`private void`	`setBmps()` sets all of the different player images
`private Bitmap`	`setBmps2(int x)` sets the player image
`private void`	`setPid()`

	Pid -> player id set the pid acording to the corrent player image
`private void`	`setUnlocked()` set witch cars are unlocked

- **Methods inherited from class java.lang.Object**
  `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

–

Field Detail

- *needToStop*
  `private boolean needToStop`

- *resumeButton*
  `private Button resumeButton`

- *carScrollLeft*
  `private Button carScrollLeft`

- *carScrollRight*
  `private Button carScrollRight`

- *background*
  `private Bitmap background`

- *title*
  `private Bitmap title`

- *pid*
  `private int pid`

  `player image pointer used to identify wich car is co rrently chosen by the player`

- *pidp*
  `private int pidp`

  `player image pointer minus`

  `points to the next car`

- *pidm*
  `private int pidm`

  `player image pointer minus`

  `points to the previous car`

- *playerImages*
  `private java.lang.String[] playerImages`

- *playerBmp*
  `private Bitmap[] playerBmp`

- *unlocked*
  `private boolean[] unlocked`

- *xPosLeft*
  `private float xPosLeft`

- *xPosRight*
  `private float xPosRight`

- *xPosMiddle*
  `private float xPosMiddle`

- *unlocktext*
  `private java.lang.String[] unlocktext`

- *unlockPaint*
  `private Paint unlockPaint`

- *screenWidth*
  `int screenWidth`

- *screenHeight*
  `int screenHeight`

–

Constructor Detail

- *OptionsView*
  `public OptionsView(Context context)`

  creates the options view

  Parameters:

  `context` -

–

Method Detail

- *setUnlocked*
  `private void setUnlocked()`

  set witch cars are unlocked

- *setPid*
  `private void setPid()`

  Pid -> player id set the pid acording to the corrent player image

- *setBmps*
  `private void setBmps()`

  sets all of the different player images

- *setBmps2*
  `private Bitmap setBmps2(int x)`

  sets the player image

Parameters:

x -

- *onDraw*
  `protected void onDraw(Canvas canvas)`

  draws the options view

  Parameters:

  `canvas` -

- *run*
  `public void run()`

  checks what buttons are being pressed

  Specified by:

  `run` in interface `java.lang.Runnable`

- *openLoginActivity*
  `private void openLoginActivity()`

  swiches to the login activity

- *resume*
  `public void resume()`

  resumes corrent activity

MusicService

2020-04-15

com.arg.hmmm.carhell

Class MusicService

- java.lang.Object
  - Service
    - com.arg.hmmm.carhell.MusicService

```
public class MusicService
extends Service
```

–

Nested Class Summary

Nested Classes

Modifier and Type	Class and Description
`class`	`MusicService.HeadsetIntentReceiver` checks for when headphones are being pluged in or out

–

Field Summary

Fields

Modifier and Type	Field and Description
`private static MediaPlayer`	`mPlayer`
`private int`	`mStartID`
`private static java.util.ArrayList<MediaPlayer>`	`musics`
`(package private) MusicService.HeadsetIntentReceiver`	`receiver`
`private static java.lang.String`	`state`

`private java.lang.String`	TAG
–	

Constructor Summary

Constructors

Constructor and Description
`MusicService()` –

Method Summary

All Methods *Static Methods* *Instance Methods* *Concrete Methods*

Modifier and Type	Method and Description
`static void`	`changeMusic(java.lang.String music)` changes wich music is corrently playing
`static void`	`createmusics(Context context)` creates the list of musics that can be played
`IBinder`	`onBind(Intent intent)`
`void`	`onCreate()` creates the music service
`void`	`onDestroy()` stops all of the music when the service is destroyed
`int`	`onStartCommand(Intent intent, int flags, int startid)`
`static void`	`pauseAll()` pauses all of the music corrently playing
`void`	`resetLowVolume()` returns the volume to normal
`void`	`setLowVolume()` sets the volume to low
`static void`	`setState(java.lang.String s)`
`static void`	`setVolume(float leftVolume, float rightVolume)` sets the music to a costume level
`private void`	`updateState(java.lang.String state)` checks if there are headphones pluged in or not, changing the volume acordingly

- Methods inherited from class java.lang.Object
  `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

–

Field Detail

- *TAG*
  `private final java.lang.String TAG`

  See Also:
  Constant Field Values

- *mPlayer*
  `private static MediaPlayer mPlayer`

- *musics*
  `private static java.util.ArrayList<MediaPlayer> musics`

- *mStartID*
  `private int mStartID`

- *state*
  `private static java.lang.String state`

- *receiver*
  `MusicService.HeadsetIntentReceiver receiver`

–

Constructor Detail

- *MusicService*
  `public MusicService()`

–

Method Detail

- *onCreate*

  `public void onCreate()`

  creates the music service

- *changeMusic*

  ```
  public static void changeMusic(java.lang.String musi
  c)
  ```

  changes wich music is corrently playing

  Parameters:

  `music` -

- *createmusics*

  `public static void createmusics(Context context)`

  creates the list of musics that can be played

  Parameters:

  `context` -

- *setLowVolume*

  `public void setLowVolume()`

  sets the volume to low

- *resetLowVolume*

  `public void resetLowVolume()`

  returns the volume to normal

- *pauseAll*

  `public static void pauseAll()`

  pauses all of the music corrently playing

- *setVolume*

  ```
  public static void setVolume(float leftVolume,
                               float rightVolume)
  ```

  sets the music to a costume level

Parameters:
`leftVolume` -
`rightVolume` -

- *onStartCommand*

```
public int onStartCommand(Intent intent,
                          int flags,
                          int startid)
```

- *onDestroy*

```
public void onDestroy()
```

  stops all of the music when the service is destroyed

- *onBind*

```
public IBinder onBind(Intent intent)
```

- *setState*

```
public static void setState(java.lang.String s)
```

- *updateState*

```
private void updateState(java.lang.String state)
```

  checks if there are headphones pluged in or not, changing the volume acordingly

MusicService.HeadsetIntentReceiver

2020-04-15

com.arg.hmmm.carhell

Class MusicService.HeadsetIntentReceiver

- java.lang.Object
  - BroadcastReceiver
    - com.arg.hmmm.carhell.MusicService.HeadsetIntentReceiver
- **Enclosing class:**
- MusicService

```
public class MusicService.HeadsetIntentReceiver
extends BroadcastReceiver
```

checks for when headphones are being pluged in or out

–

Constructor Summary

Constructors

Constructor and Description
`HeadsetIntentReceiver()`

–

Method Summary

All Methods *Instance Methods* *Concrete Methods*

Modifier and Type	Method and Description
`void`	`onReceive(Context context, Intent intent)`

- ### Methods inherited from class java.lang.Object
  `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

–

Constructor Detail

- *HeadsetIntentReceiver*
  ```
  public HeadsetIntentReceiver()
  ```

–

Method Detail

- *onReceive*
  ```
  public void onReceive(Context context,
                        Intent intent)
  ```

Input

2020-04-15

com.arg.hmmm.carhell

Class Input

- java.lang.Object
  - com.arg.hmmm.carhell.Input

```
public class Input
extends java.lang.Object
```

–

Field Summary

Fields

Modifier and Type	Field and Description
`static boolean`	`down`
`static int`	`downId`
`static <any>`	`touchPoints`
`static PointF`	`upPos`

–

Constructor Summary

Constructors

Constructor and Description
`Input()` Input is used to detect screen input from touch event

–

Method Summary

All Methods *Static Methods* *Instance Methods* *Concrete Methods*

Modifier and Type	Method and Description
`static <any>`	`getTouchPoints()` returns the points in witch the screen is touched
`boolean`	`onTouch(View v, MotionEvent event)`

adds and removes from touchpoints acording to where the screen is touched, allowing multible touch inputs in the same time

-

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

–

Field Detail

- *touchPoints*
  `public static <any> touchPoints`

- *upPos*
  `public static PointF upPos`

- *down*
  `public static boolean down`

- *downId*
  `public static int downId`

–

Constructor Detail

- *Input*
  `public Input()`

  Input is used to detect screen input from touch event

–

Method Detail

- *getTouchPoints*
  `public static <any> getTouchPoints()`

  returns the points in witch the screen is touched
  Returns:

- *onTouch*
  ```
  public boolean onTouch(View v,
                         MotionEvent event)
  ```
  adds and removes from touchpoints acording to where the screen is touched, allowing multible touch inputs in the same time

Button

2020-04-15

com.arg.hmmm.carhell

Class Button

- java.lang.Object
  - com.arg.hmmm.carhell.Button

```
public class Button
extends java.lang.Object
```

–

Field Summary

Fields

Modifier and Type	Field and Description
`private boolean`	`activate`
`private boolean`	`activateOnActionUp`
`private boolean`	`clicked`
`private float`	`height`
`private Bitmap`	`image`
`private boolean`	`pressed`
`private boolean`	`shouldBePressed`
`private java.lang.String`	`text`
`private float`	`width`
`private float`	`x`
`private float`	`y`

–

Constructor Summary

Constructors

Constructor and Description
`Button(float posX,        float posY,`

`Bitmap image,       boolean shouldBePressed,`
`boolean activateOnActionUp)`

creates a nea button with an image

`Button(float posX,       float posY,`
`float width,       float height,`
`java.lang.String text,       boolean shouldBePressed,`
`boolean activateOnActionUp)`

creates a new button with text in it

–

Method Summary

All Methods *Instance Methods* *Concrete Methods*

Modifier and Type	Method and Description
`void`	`draw(Canvas canvas)` draws the button
`Bitmap`	`getImage()` get the button's image
`private float`	`getTextSizeForHeight(Paint paint, float desiredHeight, java.lang.String text)` returns the text size to fit inside the button
`float`	`getX()` gets the button's x position
`float`	`getY()` gets the button's y position
`void`	`setText(java.lang.String text)` set the text inside the button
`boolean`	`shouldActivate()` returns true if the button should activate or not
`void`	`update()` updates the buttons state, active or not

- ### Methods inherited from class java.lang.Object

  `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

–

Field Detail

- *x*
  `private float x`

- *y*
  `private float y`

- *width*
  `private float width`

- *height*
  `private float height`

- *text*
  `private java.lang.String text`

- *activate*
  `private boolean activate`

- *clicked*
  `private boolean clicked`

- *pressed*
  `private boolean pressed`

- *shouldBePressed*
  `private boolean shouldBePressed`

- *activateOnActionUp*
  `private boolean activateOnActionUp`

- *image*
  `private Bitmap image`

–

Constructor Detail

- *Button*

```
public Button(float posX,
              float posY,
              float width,
              float height,
              java.lang.String text,
              boolean shouldBePressed,
              boolean activateOnActionUp)
```

creates a new button with text in it

Parameters:

`posX` -

`posY` -

`width` -

`height` -

`text` -

`shouldBePressed` -

`activateOnActionUp` -

- *Button*

```
public Button(float posX,
              float posY,
              Bitmap image,
              boolean shouldBePressed,
              boolean activateOnActionUp)
```

creates a nea button with an image

Parameters:

`posX` -

`posY` -

`image` -

`shouldBePressed` -

`activateOnActionUp` -

–

Method Detail

- *getImage*

```
public Bitmap getImage()
```

get the button's image

Returns:

- *getX*
  ```
  public float getX()
  ```
  gets the button's x position
  Returns:

- *getY*
  ```
  public float getY()
  ```
  gets the button's y position
  Returns:

- *setText*
  ```
  public void setText(java.lang.String text)
  ```
  set the text inside the button

- *shouldActivate*
  ```
  public boolean shouldActivate()
  ```
  returns true if the button should activate or not

- *draw*
  ```
  public void draw(Canvas canvas)
  ```
  draws the button

- *update*
  ```
  public void update()
  ```
  updates the buttons state, active or not

- *getTextSizeForHeight*
  ```
  private float getTextSizeForHeight(Paint paint,
                                    float desiredHeig
  ht,
                                    java.lang.Strin
  g text)
  ```
  returns the text size to fit inside the button

Pref

2020-04-15

com.arg.hmmm.carhell

Class Pref

- java.lang.Object
  - com.arg.hmmm.carhell.Pref

```
public class Pref
extends java.lang.Object
```

–

Field Summary

Fields

Modifier and Type	Field and Description
`private static SharedPreferences`	`pref`

–

Constructor Summary

Constructors

Constructor and Description
`Pref()` the Pref class is used to store game options after the closing the game

–

Method Summary

All Methods *Static Methods* *Concrete Methods*

Modifier and Type	Method and Description
`static void`	`createPref(Context context)` creates a new shared preferance
`static boolean`	`getBigButtons()` returs true if the big buttons option is enabled and false if not
`static`	`getHighScore()`

`float`	returns the corrent highscore
`static java.lang.String`	`getPlayerImage()` get the las set player image
`static boolean`	`getVolumeState()` returns tru if the volume is active and false if volume is turned off
`static void`	`setBigButtons(boolean b)` sets is the big buttons option is enabled or not
`static void`	`setHighScore(float score)` sets a high score
`static void`	`setPlayerImage(java.lang.String s)` sets the player's corrent image the player's image can be chosen it it saves after closing the game
`static void`	`SetVolumeState(boolean b)` sets is the volume on or off

-

  ### Methods inherited from class java.lang.Object
  `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

–

Field Detail

- *pref*
  `private static SharedPreferences pref`

–

Constructor Detail

- *Pref*
  `public Pref()`

  the Pref class is used to store game options after the closing the game

–

Method Detail

- *createPref*
  `public static void createPref(Context context)`

  creates a new shared preferance

  Parameters:

  `context` -

- *getHighScore*
  `public static float getHighScore()`

  returns the corrent highscore

  Returns:

- *setHighScore*
  `public static void setHighScore(float score)`

  sets a high score

  Parameters:

  `score` -

- *setBigButtons*
  `public static void setBigButtons(boolean b)`

  sets is the big buttons option is enabled or not

  Parameters:

  `b` -

- *getBigButtons*
  `public static boolean getBigButtons()`

  returs true if the big buttons option is enabled and false if not

  Returns:

- *setPlayerImage*
  ```
  public static void setPlayerImage(java.lang.Strin
  g s)
  ```

  sets the player's corrent image the player's image can be chosen it it saves after closing the game

  Parameters:

  `s` -

- *getPlayerImage*
  ```
  public static java.lang.String getPlayerImage()
  ```
  get the las set player image
  Returns:

- *SetVolumeState*
  ```
  public static void SetVolumeState(boolean b)
  ```
  sets is the volume on or off
  Parameters:
  b -

- *getVolumeState*
  ```
  public static boolean getVolumeState()
  ```
  returns tru if the volume is active and false if volume is turned off
  Returns:

Thing

2020-04-15

com.arg.hmmm.carhell

Class Thing

- java.lang.Object
  - com.arg.hmmm.carhell.Thing
- **All Implemented Interfaces:**
- java.lang.Runnable

  Direct Known Subclasses:

  Player, Shield

```
public class Thing
extends java.lang.Object
implements java.lang.Runnable
```

–

Field Summary

Fields

Modifier and Type	Field and Description
`protected Bitmap`	`bmp`
`protected Matrix`	`matrix`
`protected boolean`	`needToStop`
`protected Paint`	`paint`
`protected java.lang.Thread`	`thread`
`protected GameView`	`view`
`protected float`	`vx` speed on the x axis
`protected float`	`vy` speed on the y axis
`protected float`	`x`
`protected float`	`y`

–

Constructor Summary

Constructors

Constructor and Description
`Thing(float x, float y, GameView view)` creates a new Thing at (x,y) without an image
`Thing(float x, float y, GameView view, float vx, float vy, Bitmap bmp)` creates a new thing at (x,y) with a provided image

–

Method Summary

All Methods *Instance Methods* *Concrete Methods*

Modifier and Type	Method and Description
`void`	`addVx(float x)` adds to the player's horizontal speed (on the x axis)
`void`	`addVy(float x)` adds to the player's vertical speed (on the y axis)
`private void`	`checkRemove()` checks if the thing is outside the phone/game's borders, if so deletes it using remove()
`void`	`draw(Canvas canvas)` draws the thing at (x,y)
`Bitmap`	`getBmp()` returns the thing's image

`float`	`getBmpHeight()`
	return the thing's image height

`float`	`getBmpWidth()`
	return the thing's image width

`float`	`getVx()` returns the thing's horisontal speed (on the x axis)
`float`	`getVy()` returns the thing's vertical speed (on the y axis)
`float`	`getX()` returns the thing's x value

`float`	`getY()` returns the thing's y value
`private boolean`	`no()` returns true if the player needs to stop and false if he needs to continue used inside run()
`private void`	`remove()` deletes this thing from the things array in GameView
`void`	`run()` moves the thing according to its speed on the x and y axes (vx and vy)
`void`	`setBmp(Bitmap bmp)` set the thing's image
`void`	`setVx(float vx)` set the thing's horizontal speed (the the x axis)
`void`	`setVy(float vy)` set the thing's vertical speed (on the y axis)
`void`	`setX(float x)` sets the thing's x value
`void`	`setY(float y)` sets the thing's y value
`void`	`start()` start the thing's thread
`void`	`stop()` stops the thing's thread

-

Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify,
notifyAll, toString, wait, wait, wait
```

–

Field Detail

- *view*

```
protected GameView view
```

- *bmp*
  `protected Bitmap bmp`

- *matrix*
  `protected Matrix matrix`

- *paint*
  `protected Paint paint`

- *x*
  `protected float x`

- *y*
  `protected float y`

- *vx*
  `protected float vx`

  speed on the x axis

- *vy*
  `protected float vy`

  speed on the y axis

- *thread*
  `protected java.lang.Thread thread`

- *needToStop*
  `protected boolean needToStop`

–

Constructor Detail

- *Thing*
  ```
  public Thing(float x,
               float y,
               GameView view)
  ```
  creates a new Thing at (x,y) without an image

  Parameters:

  x - thing's x value

  y - thing's y value

  `view` -

- *Thing*
  ```
  public Thing(float x,
               float y,
               GameView view,
               float vx,
               float vy,
               Bitmap bmp)
  ```
  creates a new thing at (x,y) with a provided image

  Parameters:

  x - thing's x value

  y - thing's y value

  `view` -

  vx - thing's horisontal speed (on x axis)

  vy - thing's vertical speed (on y axis)

  bmp - thing's image

–

Method Detail

- *getX*
  ```
  public float getX()
  ```
  returns the thing's x value

  Returns:

- *getY*
  ```
  public float getY()
  ```
  returns the thing's y value

  Returns:

- *getVx*
  `public float getVx()`

  returns the thing's horisontal speed (on the x axis)
  Returns:

- *getVy*
  `public float getVy()`

  returns the thing's vertical speed (on the y axis)
  Returns:

- *getBmpWidth*
  `public float getBmpWidth()`

  return the thing's image width
  Returns:

- *getBmpHeight*
  `public float getBmpHeight()`

  return the thing's image height
  Returns:

- *getBmp*
  `public Bitmap getBmp()`

  returns the thing's image
  Returns:

- *setX*
  `public void setX(float x)`

  sets the thing's x value
  Parameters:
  x -

- *setY*
  `public void setY(float y)`

sets the thing's y value

Parameters:

y -

- *setVx*

  `public void setVx(float vx)`

  set the thing's horizontal speed (the the x axis)

  Parameters:

  vx -

- *setVy*

  `public void setVy(float vy)`

  set the thing's vertical speed (on the y axis)

  Parameters:

  vy -

- *setBmp*

  `public void setBmp(Bitmap bmp)`

  set the thing's image

  Parameters:

  bmp -

- *stop*

  `public void stop()`

  stops the thing's thread

- *start*

  `public void start()`

  start the thing's thread

- *addVx*

  `public void addVx(float x)`

  adds to the player's horizontal speed (on the x axis)

  Parameters:

  x -

- *addVy*
  `public void addVy(float x)`

  adds to the player's vertical speed (on the y axis)

  Parameters:

  `x` -

- *draw*
  `public void draw(Canvas canvas)`

  draws the thing at (x,y)

  Parameters:

  `canvas` -

- *checkRemove*
  `private void checkRemove()`

  checks if the thing is outside the phone/game's borders, if so deletes it using remove()

- *remove*
  `private void remove()`

  deletes this thing from the things array in GameView

- *run*
  `public void run()`

  moves the thing according to its speed on the x and y axes (vx and vy)

  Specified by:

  `run` in interface `java.lang.Runnable`

- *no*
  `private boolean no()`

  returns true if the player needs to stop and false if he needs to continue used inside run()

  Returns:

Player

2020-04-15

com.arg.hmmm.carhell

Class Player

- java.lang.Object
  - com.arg.hmmm.carhell.Thing
    - com.arg.hmmm.carhell.Player
- **All Implemented Interfaces:**
- java.lang.Runnable

```
public class Player
extends Thing
```

–

Field Summary

Fields

Modifier and Type	Field and Description
private float	angle
private boolean	isAccelerating
private boolean	isSheildActive
private float	movementFixer movementFixer is recuiered else when the car points upwords it will move to the right
private java.util.TimerTask	powerUpDuration
private Bitmap	shieldBmp
private float	speed
private float	speedFactor

-

Fields inherited from class com.arg.hmmm.carhell.Thing

`bmp, matrix, needToStop, paint, thread, view, vx, vy, x, y`

–

Constructor Summary

Constructors

Constructor and Description
`Player(int x, int y, GameView gameView, Bitmap bmp)` Creates a new player

–

Method Summary

All Methods *[Instance Methods]* *[Concrete Methods]*

Modifier and Type	Method and Description
`private void`	`activateShild()` activates the shield power-up making the player invulnerable for a couple of seconds
`void`	`brake()` reduses the playes speed faster then normal if the speed drops to 0 the player will start accelerating backwords
`private void`	`checkBoundaries()` checks if the player is touching the phone/game boundaries
`private void`	`checkImpact()` gets the things array from GameView and check for impact between the player and a thing
`void`	`draw(Canvas canvas)` draws the player and the shield (if active)
`void`	`rotateImage()` rotats the player's image depending on the player's angle
`void`	`run()` moves the player player depending on his speed, angle and is he accelerating
`void`	`setAngle(float x)` set the angle in wich the player moves
`void`	`setIsAccelerating(boolean x)` sets is the player accelerating or not
`void`	`setSpeed(float x)` sets the player's speed

`void`	`turnLeft()` turns the player left
`void`	`turnRight()` turns the player right
`private boolean`	`yes()` returns true if the player need to continue and false if he needs to stop used inside run()

- Methods inherited from class com.arg.hmmm.carhell.Thing
  `addVx, addVy, getBmp, getBmpHeight, getBmpWidth, getVx, getVy, getX, getY, setBmp, setVx, setVy, setX, setY, start, stop`

- Methods inherited from class java.lang.Object
  `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

–

Field Detail

- *isAccelerating*
  `private boolean isAccelerating`

- *speed*
  `private float speed`

- *angle*
  `private float angle`

- *movementFixer*
  `private float movementFixer`

  `movementFixer is recuiered else when the car points upwords it will move to the right`

- *speedFactor*
  `private final float speedFactor`

- *isSheildActive*
  `private boolean isSheildActive`

- *powerUpDuration*
  `private java.util.TimerTask powerUpDuration`

- *shieldBmp*
  `private Bitmap shieldBmp`

–

Constructor Detail

- *Player*

  ```
  public Player(int x,
                int y,
                GameView gameView,
                Bitmap bmp)
  ```

  Creates a new player

  Parameters:

  x - the player's x value

  y - the player's y value

  `gameView` -

  bmp - the image of the player

–

Method Detail

- *setIsAccelerating*
  `public void setIsAccelerating(boolean x)`

  sets is the player accelerating or not

  Parameters:

  x - true or false

- *setSpeed*
  `public void setSpeed(float x)`

  sets the player's speed

Parameters:

x -

- *setAngle*

  `public void setAngle(float x)`

  set the angle in wich the player moves

  Parameters:

  x -

- *turnRight*

  `public void turnRight()`

  turns the player right

- *turnLeft*

  `public void turnLeft()`

  turns the player left

- *brake*

  `public void brake()`

  reduses the playes speed faster then normal if the speed drops to 0 the player will start accelerating backwords

- *rotateImage*

  `public void rotateImage()`

  rotats the player's image depending on the player's angle

- *checkBoundaries*

  `private void checkBoundaries()`

  checks if the player is touching the phone/game boundaries

- *checkImpact*

  `private void checkImpact()`

  gets the things array from GameView and check for impact between the player and a thing

- *activateShild*
  `private void activateShild()`

  activates the shield power-up making the player invulnerable for a couple of seconds

- *draw*
  `public void draw(Canvas canvas)`

  draws the player and the shield (if active)
  Overrides:
  `draw` in class `Thing`
  Parameters:
  `canvas` -

- *run*
  `public void run()`

  moves the player player depending on his speed, angle and is he accelerating
  Specified by:
  `run` in interface `java.lang.Runnable`
  Overrides:
  `run` in class `Thing`

- *yes*
  `private boolean yes()`

  returns true if the player need to continue and false if he needs to stop used inside run()
  Returns:

PowerUp

2020-05-20

com.arg.hmmm.carhell

Class PowerUp

- java.lang.Object
  - com.arg.hmmm.carhell.Thing
    - com.arg.hmmm.carhell.PowerUp
- **All Implemented Interfaces:**
- java.lang.Runnable

```
public class PowerUp
extends com.arg.hmmm.carhell.Thing
```

–

Field Summary

Fields

Modifier and Type	Field and Description
`private java.util.TimerTask`	`secondsTask`
`private java.lang.String`	`type`

•

Fields inherited from class com.arg.hmmm.carhell.Thing

`bmp, matrix, needToStop, paint, thread, view, vx, vy, x, y`

–

Constructor Summary

Constructors

Constructor and Description
`PowerUp(float x, float y, java.lang.String type, com.arg.hmmm.carhell.GameView gameView)` creates a new power-up

–

Method Summary

All Methods *Instance Methods* *Concrete Methods*

Modifier and Type	Method and Description
`java.lang.String`	`getType()`
`private void`	`lifeTimer()` this functions has a timer that counts how long the until the PowerUp disappears the PowerUp only appears for a certain length of time
`private void`	`removePowerUp()` removes the shield

- #### Methods inherited from class com.arg.hmmm.carhell.Thing
  `addVx, addVy, draw, getBmp, getBmpHeight, getBmpWidth, getVx, getVy, getX, getY, run, setBmp, setVx, setVy, setX, setY, start, stop`

- #### Methods inherited from class java.lang.Object
  `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

–

Field Detail

- *secondsTask*
  ```
  private java.util.TimerTask secondsTask
  ```

- *type*
  ```
  private java.lang.String type
  ```

–

Constructor Detail

- *PowerUp*
  ```
  public PowerUp(float x,
                 float y,
                 java.lang.String type,
                 com.arg.hmmm.carhell.GameView gameVie
  w)
  ```

creates a new power-up

–

Method Detail

- *getType*

  public java.lang.String getType()

- *lifeTimer*

  private void lifeTimer()

  this functions has a timer that counts how long the until the PowerUp disappears the PowerUp only appears for a certain length of time

- *removePowerUp*

  private void removePowerUp()

  removes the shield

הקוד:

MainActivity

```java
package com.arg.hmmm.carhell;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.view.WindowManager;

public class MainActivity extends AppCompatActivity {
    private GameView gameView;

    /**
     * creates the main activity
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        gameView = new GameView(this);
        gameView.setOnTouchListener(new Input());
        setContentView(R.layout.activity_main);
        this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

        //music
        MusicService.changeMusic("level");

        /*
        musicServiceIntent = new Intent(getApplicationContext(),
                MusicService.class);
        startService(musicServiceIntent);*/

        //this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
        setContentView(gameView);
    }

    /**
     * stops the music when the activity stops
     */
    protected void onStop(){
        super.onStop();
        MusicService.pauseAll();
    }

    /**
     * resumes the music when the activity resumes
     */
    protected void onResume(){
        super.onResume();
        MusicService.changeMusic("level");
        gameView.resume();
    }}
```

GameView

```
package com.arg.hmmm.carhell;

import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;

import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Random;
import java.util.Timer;
import java.util.TimerTask;

import android.graphics.Color;
import android.graphics.Paint;
import android.view.View;

class GameView extends View implements Runnable {
    private Player player;
    private ArrayList<Thing> things;
    //private Shield shield;
    private int screenWidth;
    private int screenHeight;
    private Bitmap background;
    private float seconds;
    private Paint paint;
    private TimerTask summonTearTask;
    private int summonTearTaskPeriod;
    private TimerTask secondsTask;
    private TimerTask summonPowerUpTask;
    private boolean needToStop;
    private boolean isdead;
    private boolean paused;
    private Bitmap ripbmp;
    private Bitmap filterbmp;
    private Paint filterpaint;
    //print 1.00
    private DecimalFormat df2;

    private Thing testPixle;

    //buttons
    private Button gaspedalS;
    private Button breakpedalS;
    private Button leftarrowS;
    private Button rightarrowS;
    private Button gaspedalL;
    private Button breakpedalL;
    private Button leftarrowL;
    private Button rightarrowL;
    private Button restart;
    private Button returnToMainButton;
    private Button resumeButton;
    private Button pauseButton;
    private Button buttonSizeChange;
    private Button volumeOnButton;
    private Button volumeOffButton;


    /**
     * starts the game
     */
    public GameView(Context context) {
        super(context);

        screenWidth = getResources().getDisplayMetrics().widthPixels;
        screenHeight = getResources().getDisplayMetrics().heightPixels;

        /* this doesnt work, screen hieght and width are still 0 when out of the while.
           writing a do while makes it crash becose its infinite
           help
        while(screenWidth==0 || screenHeight==0){
            screenWidth = getResources().getDisplayMetrics().widthPixels;
            screenHeight = getResources().getDisplayMetrics().heightPixels;
        }
        */



        testPixle = new
Thing(0,0,this,0,0,Bitmap.createScaledBitmap(BitmapFactory.decodeResource(this.getResources(),R.d
rawable.whitepixle), screenWidth/35, screenHeight/12, false));




        /*
        player = new
Player(screenWidth/2,screenHeight/2,this,BitmapFactory.decodeResource(this.getResources(),R.drawa
ble.car));
        //player.setBmp(Bitmap.createScaledBitmap(player.getBmp(), ScreenWidth/35, ScreenHeight/12,
false));
        player.setBmp(Bitmap.createScaledBitmap(player.getBmp(), screenWidth/35, screenHeight/12,
false));
        //to stop player from spawning at 0,0
        player.setX(screenWidth/2);
        player.setY(screenHeight/2);*/

        createPlayer();

        things = new ArrayList<Thing>();
        //summonTear();

        background = BitmapFactory.decodeResource(this.getResources(),R.drawable.background45);
        background = Bitmap.createScaledBitmap(background, screenWidth, screenHeight, false);

        filterbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.pausfilter);
        filterbmp.setHasAlpha(true);
        filterpaint = new Paint();
        filterpaint.setAlpha(150);
        filterbmp = Bitmap.createScaledBitmap(background, screenWidth, screenHeight, false);

        ripbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.letterf);
        ripbmp = Bitmap.createScaledBitmap(ripbmp, screenWidth/5, 2*screenHeight/5, false);

        //bigButtons = false;

        //create buttons
        createButtons();

        paint = new Paint();
        paint.setColor(Color.RED);
        paint.setTextSize(screenHeight*0.05f);
        seconds = 0;

        //summonTears = true;
        startTearTimer();
        //250
        startPowerUpTimer();


        df2 = new DecimalFormat("#.##");
        startSecondsTimer();

        isdead=false;
        paused = false;

        needToStop = false;
        Thread thread = new Thread(this);
        thread.start();
    }

    /**
     * draws everything in the game
     */
    protected void onDraw(Canvas canvas){
        super.onDraw(canvas);

        canvas.drawBitmap(background,0,0,null);
        canvas.drawText(""+df2.format(seconds),screenWidth/50,screenHeight/12,paint);

        //testPixle.draw(canvas);
        //player.drawMatrix(canvas);
        player.draw(canvas);

        for (int i=0; i<things.size();i++) {
            things.get(i).draw(canvas);
        }

        if(Pref.getBigButtons()){
            gaspedalL.draw(canvas);
            breakpedalL.draw(canvas);
            leftarrowL.draw(canvas);
            rightarrowL.draw(canvas);
        } else{
            gaspedalS.draw(canvas);
            breakpedalS.draw(canvas);
            leftarrowS.draw(canvas);
            rightarrowS.draw(canvas);
            }
    if(isdead){
        canvas.drawBitmap(filterbmp,0,0,filterpaint);
        canvas.drawBitmap(ripbmp,screenWidth/2 - ripbmp.getWidth()/2,screenHeight/10,null);
        restart.draw(canvas);
        returnToMainButton.draw(canvas);

    } else{
        if(paused) {
            canvas.drawBitmap(filterbmp,0,0,filterpaint);
            resumeButton.draw(canvas);
            returnToMainButton.draw(canvas);
            buttonSizeChange.draw(canvas);
            if(Pref.getVolumeState())
                volumeOnButton.draw(canvas);
            else
                volumeOffButton.draw(canvas);
        } else{
            pauseButton.draw(canvas);
        }






    }
    invalidate();
}

/**
 * removes thing from the array
 */
public void removeThing(Thing thing){
    if(thing != null)
        things.remove(thing);
}

/**
 * adds thing to the array
 */
public void addThing(Thing thing){
    things.add(thing);
}

/**
 * returns things array
 */
public ArrayList<Thing> getThings(){
    return things;
}

/**
 * returns the test pixle (used for testing)
 */
public Thing getTestPixle(){
    return testPixle;
}
```

```
/**
 * returns the player
 */
public Player getPlayer() {
    return player;
}

/**
 * returns the screen width
 */
public int getScreenWidth() {
    return screenWidth;
}

/**
 * returns the screen height
 */
public int getScreenHeight() {
    return screenHeight;
}

/**
 * generates the player
 */
private void createPlayer(){
    Bitmap  tbmp;
    switch(Pref.getPlayerImage()){
        case "yellow":
            tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.car);
            break;
        case "green":
            tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.cargreen);
            break;
        case "gray":
            tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.cargray);
            break;
        case "blue":
            tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.carinverted);
            break;
        case "pink":
            tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.carpink);
            break;
        case "orange":
            tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.carorange);
            break;
        case "white":
            tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.carwhite);
            break;
        case "secret":
            tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.secretskin);
            break;
        default:
            tbmp= BitmapFactory.decodeResource(this.getResources(),R.drawable.whitepixle);
    }

    player = new Player(screenWidth/2,screenHeight/2,this, tbmp);
```

```
        //player.setBmp(Bitmap.createScaledBitmap(player.getBmp(), ScreenWidth/35, ScreenHeight/12,
false));
        player.setBmp(Bitmap.createScaledBitmap(player.getBmp(), screenWidth/35, screenHeight/12,
false));
        //to stop player from spawning at 0,0
        player.setX(screenWidth/2);
        player.setY(screenHeight/2);
    }






    //block of cheese
    /**
     * generates all of the buttons
     */
    private void createButtons(){
        //affects menu buttons
        final float buttonXpos = screenWidth/3f;
        final float buttonYpos = screenHeight*0.6f;
        final float distanceBetweenButtons = screenHeight*0.2f;
        final float buttonWidth = screenWidth/3f;
        final float buttonHieght = screenHeight/10f;

        gaspedalS = new Button(screenWidth-
(11*screenWidth/100),screenHeight/2,Bitmap.createScaledBitmap(BitmapFactory.decodeResource(thi
s.getResources(),R.drawable.gaspedal), screenWidth/12, screenHeight*3/10, false),true,false);
        breakpedalS = new
Button(screenWidth*86/100,screenHeight*81/100,Bitmap.createScaledBitmap(BitmapFactory.decode
Resource(this.getResources(),R.drawable.brakepedal), screenWidth*12/100, screenHeight*15/100,
false),true,false);
        leftarrowS = new
Button(screenWidth/100,20*screenHeight/24,Bitmap.createScaledBitmap(BitmapFactory.decodeReso
urce(this.getResources(),R.drawable.leftarrow), screenWidth/7, screenHeight/7, false),true,false);
        rightarrowS = new
Button(15*screenWidth/100,20*screenHeight/24,Bitmap.createScaledBitmap(BitmapFactory.decodeR
esource(this.getResources(),R.drawable.rightarrow), screenWidth/7, screenHeight/7, false),true,false);

        gaspedalL = new Button(screenWidth-
(11*screenWidth/100),screenHeight/3,Bitmap.createScaledBitmap(BitmapFactory.decodeResource(thi
s.getResources(),R.drawable.gaspedal), screenWidth/10, 2*screenHeight/5, false),true,false);
        breakpedalL = new Button(screenWidth-
(15*screenWidth/100),18*screenHeight/24,Bitmap.createScaledBitmap(BitmapFactory.decodeResourc
e(this.getResources(),R.drawable.brakepedal), 14*screenWidth/100, screenHeight/5, false),true,false);
        leftarrowL = new
Button(screenWidth/100,18*screenHeight/24,Bitmap.createScaledBitmap(BitmapFactory.decodeReso
urce(this.getResources(),R.drawable.leftarrow), screenWidth/5, screenHeight/5, false),true,false);
        rightarrowL = new
Button(22*screenWidth/100,18*screenHeight/24,Bitmap.createScaledBitmap(BitmapFactory.decodeR
esource(this.getResources(),R.drawable.rightarrow), screenWidth/5, screenHeight/5, false),true,false);

        pauseButton = new Button(screenWidth- screenWidth/50f -
screenHeight/11f,screenHeight/50,Bitmap.createScaledBitmap(BitmapFactory.decodeResource(this.ge
tResources(),R.drawable.pausewhite), screenHeight/10, screenHeight/10, false),true,true);
        restart = new Button(buttonXpos, buttonYpos,buttonWidth, buttonHieght, "ReStArT",true,true);
        returnToMainButton = new Button(buttonXpos,
buttonYpos+distanceBetweenButtons,buttonWidth, buttonHieght, "Main",true,true);
        resumeButton = new Button(buttonXpos, buttonYpos,buttonWidth, buttonHieght,
"rEsuMe",true,true);

        volumeOnButton = new Button(screenWidth- screenWidth/50f -
screenHeight/11f,screenHeight/50,
Bitmap.createScaledBitmap(BitmapFactory.decodeResource(this.getResources(),R.drawable.volumeon
), screenHeight/10, screenHeight/10, false),true,true);
        volumeOffButton = new Button(screenWidth- screenWidth/50f -
screenHeight/11f,screenHeight/50,
Bitmap.createScaledBitmap(BitmapFactory.decodeResource(this.getResources(),R.drawable.volumeoff
), screenHeight/10, screenHeight/10, false),true,true);

        String tText = "Button Size: S";
        if(Pref.getBigButtons())
            tText = "Button Size: L";
        buttonSizeChange = new Button(buttonXpos, buttonYpos-distanceBetweenButtons,buttonWidth,
buttonHieght, tText,true,true);


    }

    /**
     * start the tear timer
     * tear timer is responsible for the tear spawn rate
     */
    private void startTearTimer(){
        summonTearTaskPeriod = 400;
        Timer summonTearTimer = new Timer();
        summonTearTask = new TimerTask() {
            @Override
            public void run() {
                //add here something to stop, if()
                if(seconds > 1){
                    summonTear();
                    if(400-seconds/20>=50)
                        summonTearTaskPeriod = 400-(int)(seconds/20);
                }
            }
        };
        summonTearTimer.scheduleAtFixedRate(summonTearTask,0,summonTearTaskPeriod);
    }

    /**
     * start the power up timer
     * power up timer responsible for the  power up spawn rate
     */
    private void startPowerUpTimer(){
        final Random rnd = new Random();
        final GameView tview = this;
        Timer summonPowerUpTimer = new Timer();
        summonPowerUpTask = new TimerTask() {
            @Override
            public void run() {
                //add here something to stop, if()
                float tx = 0, ty = 0;
                tx = rnd.nextInt(screenWidth);
                ty = rnd.nextInt(screenHeight);

                //things.add(new PowerUp(tx,ty,"shield", tview));
                things.add(new PowerUp(screenWidth/2,screenHeight/2,"shield", tview));
            }
        };
        summonPowerUpTimer.scheduleAtFixedRate(summonPowerUpTask,/*12*1000*/0,1000*15);
    }

    /**
     * start the seconds timer
     * seconds timer counts the seconds from the  start of the run
     */
    private void startSecondsTimer(){
        Timer secondsTimer = new Timer();
        secondsTask = new TimerTask() {
            @Override
            public void run() {
                //add here something to stop, if()
                seconds=seconds+0.01f;
            }
        };
        secondsTimer.scheduleAtFixedRate(secondsTask,0,10);
    }

    /**
     * summons a tear
     */
    public void summonTear(){
        float tx=0, ty=0 , tvx=0, tvy=0, speed = 0;
        Bitmap tbmp =
Bitmap.createScaledBitmap(BitmapFactory.decodeResource(this.getResources(),R.drawable.whitepixl
e), screenHeight/12, screenHeight/12, false);
        Random rnd = new Random();

        //tr -> tear difficulty
        int tr = 1;
        if(seconds>=25)
            tr=3;
        else if(seconds >=10)
            tr=2;

        switch (rnd.nextInt(tr)){//chooses the type of projectile
            case 0: //random diraction random spawn
                tbmp =
Bitmap.createScaledBitmap(BitmapFactory.decodeResource(this.getResources(),R.drawable.tear),
screenHeight/12, screenHeight/12, false);
                float angle = (rnd.nextInt((135-45)+1)+45);    // rand.nextInt((max - min) + 1) + min;
                //0- left, 1- up, 2- right, 3- down
                switch(rnd.nextInt(4)) {
                    case 0:
                        tx = -tbmp.getWidth();
                        ty = rnd.nextInt(screenHeight-tbmp.getHeight());
                        angle -= 90;
                        break;
                    case 1:
                        ty = -tbmp.getHeight();
                        tx = rnd.nextInt(screenWidth-tbmp.getWidth());
                        angle*= -1f;
                        break;
                    case 2:
                        tx = screenWidth;
                        ty = rnd.nextInt(screenHeight-tbmp.getHeight());
                        angle += 90;
                        break;
                    case 3:
                        ty = screenHeight;
                        tx = rnd.nextInt(screenWidth-tbmp.getWidth());
                        //angle stays the same
                        break;
                }

                speed = screenWidth/200f;
                angle = (float)Math.toRadians(angle);
                tvx = (float)(speed*Math.cos(angle));
                tvy = (float)(speed*Math.sin(angle));
                break;
            case 1: //flys to the direction of the player
                tbmp =
Bitmap.createScaledBitmap(BitmapFactory.decodeResource(this.getResources(),R.drawable.pinktear),
screenHeight/12, screenHeight/12, false);
                //0- left, 1- up, 2- right, 3- down
                switch(rnd.nextInt(4)) {
                    case 0:
                        tx = -tbmp.getWidth();
                        ty = rnd.nextInt(screenHeight - tbmp.getHeight());
                        break;
                    case 1:
                        ty = -tbmp.getHeight();
                        tx = rnd.nextInt(screenWidth - tbmp.getWidth());
                        break;
                    case 2:
                        tx = screenWidth;
                        ty = rnd.nextInt(screenHeight - tbmp.getHeight());
                        break;
                    case 3:
                        ty = screenHeight;
                        tx = rnd.nextInt(screenWidth-tbmp.getWidth());
                        break;
                }
                float dis = (float)Math.sqrt(Math.pow(player.getX() - tx, 2) + Math.pow(player.getY() - ty,
2));
                speed = screenWidth/300f;
                float poo = speed / dis;
                tvx = (player.getX() - tx) * poo;
                tvy = (player.getY() - ty) * poo;
                break;
            case 2: //same x/y moves in strait line(up/down or left/right
                tbmp =
Bitmap.createScaledBitmap(BitmapFactory.decodeResource(this.getResources(),R.drawable.greentear
), screenHeight/12, screenHeight/12, false);
                speed = screenWidth/ 150f;
                //0- left, 1- up, 2- right, 3- down
                switch(rnd.nextInt(4)) {
                    case 0:
                        tx = -tbmp.getWidth();
                        ty = rnd.nextInt(screenHeight-tbmp.getHeight());
                        tvx = speed;
                        break;
                    case 1:
                        ty = -tbmp.getHeight();
                        tx = rnd.nextInt(screenWidth-tbmp.getWidth());
                        tvy = speed;
                        break;
                    case 2:
                        tx = screenWidth;
                        ty = rnd.nextInt(screenHeight-tbmp.getHeight());
                        tvx = -speed;
                        break;
                    case 3:
                        ty = screenHeight;
                        tx = rnd.nextInt(screenWidth-tbmp.getWidth());
                        tvy = -speed;
                        break;
                }
                break;
        }



        things.add(new Thing(tx,ty,this,tvx,tvy, tbmp));

    }

    /**
     * swiches to loose screan
     */
    public void looseScrean(){
        summonTearTask.cancel();
        summonPowerUpTask.cancel();
        secondsTask.cancel();
        player.stop();
        isdead = true;
    }

    /**
     * switches to pause screen
     */
    private void pauseScrean(){
        summonTearTask.cancel();
        summonPowerUpTask.cancel();
        secondsTask.cancel();
        player.stop();
        for (Thing thing :
                things) {
            thing.stop();
        }
        paused = true;
    }

    /**
```

```
 * stops  pauses screen and returns to the game
 */
private void stopPause(){
    startSecondsTimer();
    startTearTimer();
    player.start();
    for (Thing thing :
            things) {
        thing.start();
    }
    paused = false;
}

/**
 * checks if any of the buttons have been pressed
 */
@Override
public void run() {
    while(!needToStop){
        //this is a bad idea
        //summonTear();

        if(!isdead) {
            if(seconds > Pref.getHighScore()){
                Pref.setHighScore(seconds);
            }
            if(paused){
                resumeButton.update();
                if(resumeButton.shouldActivate()){
                    stopPause();
                }
                returnToMainButton.update();
                if(returnToMainButton.shouldActivate()){
                    returnToLogin();
                }
                buttonSizeChange.update();
                if(buttonSizeChange.shouldActivate()){
                    if(Pref.getBigButtons()) {
                        buttonSizeChange.setText("Button Size: S");
                        Pref.setBigButtons(false);
                    } else {
                        buttonSizeChange.setText("Button Size: L");
                        Pref.setBigButtons(true);
                    }
                }

                if(Pref.getVolumeState()){
                    volumeOnButton.update();
                    if(volumeOnButton.shouldActivate()){
                        Pref.SetVolumeState(false);
                    }
                }
                else{
                    volumeOffButton.update();
                    if(volumeOffButton.shouldActivate()){
                        Pref.SetVolumeState(true);
                    }
                }
            } else{
                if(Pref.getBigButtons()) {
                    gaspedalL.update();
                    if (gaspedalL.shouldActivate()) {
                        player.setIsAccelerating(true);
                    } else
                        player.setIsAccelerating(false);

                    rightarrowL.update();
                    if (rightarrowL.shouldActivate())
                        player.turnRight();

                    leftarrowL.update();
                    if (leftarrowL.shouldActivate())
                        player.turnLeft();

                    breakpedalL.update();
                    if (breakpedalL.shouldActivate())
                        player.brake();
                } else{
                    gaspedalS.update();
                    if (gaspedalS.shouldActivate()) {
                        player.setIsAccelerating(true);
                    } else
                        player.setIsAccelerating(false);

                    rightarrowS.update();
                    if (rightarrowS.shouldActivate())
                        player.turnRight();

                    leftarrowS.update();
                    if (leftarrowS.shouldActivate())
                        player.turnLeft();

                    breakpedalS.update();
                    if (breakpedalS.shouldActivate())
                        player.brake();
                }

            pauseButton.update();
            if(pauseButton.shouldActivate())
                pauseScrean();
        }} else{
            restart.update();
            if(restart.shouldActivate()){
                System.out.println("restart");
                needToStop = true;

                /*
                Intent intent = new Intent(getContext(), MainActivity.class);
                getContext().startActivity(intent);*/

                restartView();

            }
```

```
                returnToMainButton.update();
                if(returnToMainButton.shouldActivate()){
                    returnToLogin();
                }

            }



            try {
                Thread.sleep(15);
            } catch (Exception e){
                e.printStackTrace();
            }
        }
    }

    /**
     * returns to login screen
     */
    private  void returnToLogin(){
        needToStop = true;
        Intent intent = new Intent(getContext(), LoginActivity.class);
        getContext().startActivity(intent);
    }

    /**
     * resumes the thread
     */
    public void resume(){
        if (needToStop) {
            needToStop = false;
            Thread thread = new Thread(this);
            thread.start();
        }
    }

    /**
     * restart the run, restarting the game
     */
    private void restartView(){
        seconds = 0;
        player.setSpeed(0);
        player.setY(screenHeight/2f);
        player.setX(screenWidth/2f);
        player.setAngle(0);

        seconds = 0;
        isdead = false;
        needToStop = false;
        player.start();
        things.clear();
        startSecondsTimer();
        startTearTimer();
        startPowerUpTimer();
    }
}
```

LoginActivity

```
package com.arg.hmmm.carhell;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.view.WindowManager;

public class LoginActivity extends AppCompatActivity {
    private LoginView loginView;
    private Intent musicServiceIntent;

    /**
     * creates the login activity
     * login activity is the first screen of the app
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        loginView = new LoginView(this);
        loginView.setOnTouchListener(new Input());
        setContentView(R.layout.activity_main);
        this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

        Pref.createPref(this);
        //music
        MusicService.createmusics(this);
        musicServiceIntent = new Intent(getApplicationContext(),
                com.arg.hmmm.carhell.MusicService.class);
        startService(musicServiceIntent);
        MusicService.changeMusic("menu");
        //this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
        setContentView(loginView);
    }

    /**
     * stops the music when the activity stops
     */
    protected void onStop(){
        super.onStop();
        MusicService.pauseAll();
    }

    /**
     * resumes the music when the activity resumes
     */
    protected void onResume(){
        super.onResume();
        MusicService.changeMusic("menu");
        loginView.resume();}}
```

LoginView

```
package com.arg.hmmm.carhell;

import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.view.View;

public class LoginView extends View implements Runnable {
   private Button playButton, optionsButton;
   private boolean needToStop2=true;
   private Bitmap title, background;
   int screenWidth,screenHeight;

   /**
    * creates a login view, in the login  the user choses if to start the game or to go to options
    */
   public LoginView(Context context) {
      super(context);


      screenWidth = getResources().getDisplayMetrics().widthPixels;
      screenHeight = getResources().getDisplayMetrics().heightPixels;

      playButton = new Button(screenWidth/3, screenHeight*0.4f,screenWidth/3, screenHeight/10,
"PlaY",true,true);
      optionsButton = new Button(screenWidth/3, screenHeight*0.6f,screenWidth/3, screenHeight/10,
"OpTiOnS",true,true);


      title =
Bitmap.createScaledBitmap(BitmapFactory.decodeResource(this.getResources(),R.drawable.titleone),
screenWidth/2, screenHeight/5, false);

      background =
Bitmap.createScaledBitmap(BitmapFactory.decodeResource(this.getResources(),R.drawable.backgrou
nd45), screenWidth, screenHeight, false);

      resume();
   }

   /**
    * draws everything in the view
    */
   protected void onDraw(Canvas canvas){
      super.onDraw(canvas);

      canvas.drawBitmap(background,0,0,null);
      canvas.drawBitmap(title,screenWidth/4,screenHeight*0.1f,null);

      playButton.draw(canvas);
      optionsButton.draw(canvas);

      invalidate();
   }

   /**
    * checks if any of the buttons are pressed
    */
   @Override
   public void run() {
      while(!needToStop()){

          playButton.update();
          if(playButton.shouldActivate()){
              openGameActivity();
          }

          optionsButton.update();
          if(optionsButton.shouldActivate()){
              openOptionsActivity();
          }
      }
   }

   /**
    * swiches to the game activity
    */
   private void openGameActivity(){
      needToStop2 = true;
      Intent intent = new Intent(getContext(), MainActivity.class);
      getContext().startActivity(intent);
   }

   /**
    * swiches to the options activity
    */
   private void openOptionsActivity(){
      needToStop2 = true;
      Intent intent = new Intent(getContext(), OptionsActivity.class);
      getContext().startActivity(intent);
   }

   /**
    * checks if the view needs to stop or not, used in run()
    */
   private boolean needToStop(){
      return needToStop2;
   }

   /**
    * resumes the view
    */
   public void resume(){
      if (needToStop2) {
          needToStop2 = false;
          Thread thread = new Thread(this);
          thread.start();
      }
   }
}
```

OptionsActivity

```java
package com.arg.hmmm.carhell;

import android.content.Context;
import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.view.WindowManager;

import androidx.appcompat.app.AppCompatActivity;

public class OptionsActivity extends AppCompatActivity {
    private OptionsView optionsView;

    /**
     * start the options activity
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        optionsView = new OptionsView(this);
        optionsView.setOnTouchListener(new Input());
        setContentView(R.layout.activity_main);
        this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);


        MusicService.changeMusic("menu");

        //this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
        setContentView(optionsView);

    }

    /**
     * stops the music when the activity stops
     */
    protected void onStop(){
        super.onStop();
        MusicService.pauseAll();
    }

    /**
     * resumes the music when the activity resumes
     */
    protected void onResume(){
        super.onResume();
        MusicService.changeMusic("menu");
        optionsView.resume();
    }
}
```

OptionsView

```
package com.arg.hmmm.carhell;

import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.view.View;

public class OptionsView  extends View implements Runnable  {
    private boolean needToStop=true;
    private Button resumeButton, carScrollLeft, carScrollRight;
    private Bitmap background, title;
    private int pid; //Player Image pointer
    private int pidp, pidm;//pid plus, pid minus
    private String[] playerImages;
    private Bitmap[] playerBmp;
    private boolean[] unlocked;
    private float xPosLeft, xPosRight, xPosMiddle;
    private String[] unlocktext;
    private Paint unlockPaint;

    int screenWidth,screenHeight;

    /**
     * creates the options view
     */
    public OptionsView(Context context) {
        super(context);

        screenWidth = getResources().getDisplayMetrics().widthPixels;
        screenHeight = getResources().getDisplayMetrics().heightPixels;

        resumeButton = new Button(screenWidth/3f, screenHeight*0.7f,screenWidth/3f, screenHeight/10f,
"ReSUmE",true,true);
        title =
Bitmap.createScaledBitmap(BitmapFactory.decodeResource(this.getResources(),R.drawable.optionstit
le), screenWidth/2, screenHeight/5, false);
        background =
Bitmap.createScaledBitmap(BitmapFactory.decodeResource(this.getResources(),R.drawable.backgrou
nd45), screenWidth, screenHeight, false);

        carScrollLeft = new
Button(screenHeight*0.3f,screenHeight*0.4f,Bitmap.createScaledBitmap(BitmapFactory.decodeResou
rce(this.getResources(),R.drawable.bluearrowleft), screenWidth/8, screenHeight/7, false),true,true);
        carScrollRight = new
Button(screenWidth*0.7f,screenHeight*0.4f,Bitmap.createScaledBitmap(BitmapFactory.decodeResou
rce(this.getResources(),R.drawable.bluearrowright), screenWidth/8, screenHeight/7, false),true,true);

        playerImages = new String[8];
        playerImages[0] = "yellow";
        playerImages[1] = "green";
        playerImages[2] = "gray";
        playerImages[3] = "blue";
        playerImages[4] = "pink";
        playerImages[5] = "orange";
        playerImages[6] = "white";
        playerImages[7] = "secret";

        unlocktext = new String[playerImages.length-3];
        unlocktext[0] = "Unlocked by playing a game";
        unlocktext[1] = "Unlocked by surviving 10 seconds";
        unlocktext[2] = "Unlocked by surviving 17.5 seconds";
        unlocktext[3] = "Unlocked by surviving 25 seconds";
        unlocktext[4] = "Unlocked by surviving 420 seconds";


        unlocked = new boolean[playerImages.length];
        setUnlocked();
        /*
        for(int i = 0; i <3; i++)
            unlocked[i] = true;
        for(int i=3; i< 3+Pref.getUnlockLevel(); i++)
            if(i<unlocked.length)
                unlocked[i] = true;*/
        setPid();
        playerBmp = new Bitmap[playerImages.length];
        setBmps();

        xPosLeft = carScrollLeft.getX() + carScrollLeft.getImage().getWidth() +
playerBmp[0].getWidth()*2;
        xPosMiddle = (carScrollLeft.getX() + carScrollLeft.getImage().getWidth() +
carScrollRight.getX())/2;
        xPosRight = carScrollRight.getX() - playerBmp[0].getWidth()*3;

        unlockPaint = new Paint();
        unlockPaint.setColor(Color.WHITE);
        unlockPaint.setTextSize(screenHeight*0.05f);

        resume();
    }

    /**
     * set witch cars are unlocked
     */
    private void setUnlocked(){
        for(int i = 0; i <3; i++)
            unlocked[i] = true;
        unlocked[3] = (Pref.getHighScore()>=1);
        unlocked[4] = (Pref.getHighScore()>=10);
        unlocked[5] = (Pref.getHighScore()>=17.5);
        unlocked[6] = (Pref.getHighScore()>=25);
        unlocked[7] = (Pref.getHighScore()>=420);
    }

    /**
     * Pid -> player id
     * set the pid acording to the corrent player image
     */
```

```
private void setPid(){
    switch(Pref.getPlayerImage()){
        case "green":
            pid = 1;
            break;
        case "gray":
            pid = 2;
            break;
        case "blue":
            pid = 3;
            break;
        case "pink":
            pid = 4;
            break;
        case "orange":
            pid = 5;
            break;
        case "white":
            pid = 6;
            break;
        case "secret":
            pid = 7;
            break;
        default:
            pid = 0;
    }
    pidm=pid-1;
    pidp=pid+1;
    if(pidm < 0)
        pidm = playerImages.length-1;
    if(pidp == playerImages.length)
        pidp = 0;
}

/**
 * sets all of the different player images
 */
private void setBmps(){
    for(int i=0; i<playerImages.length; i++)
        playerBmp[i] = setBmps2(i);
}

/**
 * sets the player image
 */
private Bitmap setBmps2(int x){

    Bitmap tbmp;

    if(!unlocked[x])
        tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.lock);
    else{
    switch(playerImages[x]){
        case "yellow":
            tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.car);
            break;
        case "green":
            tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.cargreen);
            break;
        case "gray":
            tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.cargray);
            break;
        case "blue":
            tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.carinverted);
            break;
        case "pink":
            tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.carpink);
            break;
        case "orange":
            tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.carorange);
            break;
        case "white":
            tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.carwhite);
            break;
        case "secret":
            tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.secretskin);
            break;
        default:
            tbmp = BitmapFactory.decodeResource(this.getResources(),R.drawable.whitepixle);
    }}

    tbmp = Bitmap.createScaledBitmap(tbmp, screenWidth/35, screenHeight/12, false);
    return tbmp;
  }

  /**
   * draws the options view
   */
  protected void onDraw(Canvas canvas){
    super.onDraw(canvas);

    canvas.drawBitmap(background,0,0,null);
    canvas.drawBitmap(title,screenWidth/4,screenHeight*0.1f,null);


    resumeButton.draw(canvas);
    carScrollLeft.draw(canvas);
    carScrollRight.draw(canvas);

    canvas.drawBitmap(playerBmp[pidm], xPosLeft,carScrollLeft.getY(),null);
    canvas.drawBitmap(playerBmp[pid], xPosMiddle,carScrollLeft.getY(),null);
    canvas.drawBitmap(playerBmp[pidp], xPosRight, carScrollRight.getY(),null);

    if(pid>=3)
        canvas.drawText(unlocktext[pid-3],xPosMiddle-unlockPaint.measureText(unlocktext[pid-
3])/2,carScrollLeft.getY()-unlockPaint.getTextSize(),unlockPaint);

    invalidate();}
  /**
   * checks what buttons are being pressed
   */
  @Override
  public void run() {
    while(!needToStop){
        resumeButton.update();
        if(resumeButton.shouldActivate()){
            if(unlocked[pid])
                Pref.setPlayerImage(playerImages[pid]);
            openLoginActivity();
        }

        carScrollLeft.update();
        if(carScrollLeft.shouldActivate()){
            pid--;
            if(pid<0)
                pid = playerImages.length-1;
            pidp= pid+1;
            pidm=pid-1;
            if(pidp >= playerImages.length)
                pidp = 0;
            if(pidm < 0)
                pidm = playerImages.length-1;
        }
        carScrollRight.update();
        if(carScrollRight.shouldActivate()){
            pid++;
            if(pid == playerImages.length)
                pid = 0;
            pidm=pid-1;
            pidp=pid+1;
            if(pidm < 0)
                pidm = playerImages.length-1;
            if(pidp >= playerImages.length)
                pidp = 0;
        }

        try {
            Thread.sleep(15);
        } catch (Exception e){
            e.printStackTrace();
        }

    }
}
/**
 * swiches to the login activity
 */
private void openLoginActivity(){
    needToStop = true;
    Intent intent = new Intent(getContext(), LoginActivity.class);
    getContext().startActivity(intent);
}
/**
 * resumes corrent activity
 */
public void resume(){
    if (needToStop) {
        needToStop = false;
        Thread thread = new Thread(this);
        thread.start();}}}
```

Input

```
package com.arg.hmmm.carhell;

import android.content.Context;
import android.graphics.PointF;
import android.util.SparseArray;
import android.view.MotionEvent;
import android.view.View;

import java.util.ArrayList;
import java.util.List;

public class Input implements View.OnTouchListener {
    public static SparseArray<PointF> touchPoints;
    /*public static float x = 0;
    public static float y = 0;*/
    public static PointF upPos;
    public static boolean down = false;
    public static int downId = -1;

    /**
     * Input is used to detect screen input from touch event
     */
    public Input() {
        touchPoints = new SparseArray<>();
    }

    /**
     * returns the points in witch the screen is touched
     */
    public static SparseArray<PointF> getTouchPoints() {
        if (touchPoints == null)
            return new SparseArray<>();
        return touchPoints.clone();
    }

    /**
     * adds and removes from touchpoints acording to where the screen is touched, allowing multible
touch inputs in the same time
     */
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        /*x = event.getX();
        y = event.getY();*/

        int pointerIndex = event.getActionIndex();
        int pointerId = event.getPointerId(pointerIndex);
        int maskedAction = event.getActionMasked();

        upPos = null;
        downId = -1;
        down = false;
        switch(maskedAction){
            case MotionEvent.ACTION_DOWN:
            case MotionEvent.ACTION_POINTER_DOWN:   //ACTION_POINTER_DOWN is used for
multiple touch inputs
```

```
                PointF f = new PointF();
                f.x = event.getX(pointerIndex);
                f.y = event.getY(pointerIndex);
                touchPoints.put(pointerId, f);
                down = true;
                downId = pointerId;
                break;
            case MotionEvent.ACTION_MOVE:/*
                for (int size = event.getPointerCount(), i = 0; i < size; i++) {
                    PointF point = touchPoints.get(event.getPointerId(i));
                    if (point != null) {
                        point.x = event.getX(i);
                        point.y = event.getY(i);
                    }
                }*/
                break;
            case MotionEvent.ACTION_UP:
                upPos = new PointF(event.getX(), event.getY());
                touchPoints.remove(pointerId);
                break;
            case MotionEvent.ACTION_POINTER_UP:
            case MotionEvent.ACTION_CANCEL:
                touchPoints.remove(pointerId);
                break;
        }
        return true;
    }
}
```

Button

```
package com.arg.hmmm.carhell;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.PointF;
import android.graphics.Rect;
import android.util.SparseArray;

public class Button {
   private float x, y, width, height;
   private String text;
   private boolean activate = false, clicked = false, pressed = false, shouldBePressed,
activateOnActionUp;
   private Bitmap image;

   /**
    * creates a new button with text in it
    */
   public Button(float posX, float posY, float width, float height, String text, boolean shouldBePressed,
boolean activateOnActionUp) {
       this.x = posX;
       this.y = posY;
       this.width = width;
       this.height = height;
       this.text = text;
       this.image = null;
       this.shouldBePressed = shouldBePressed;
       this.activateOnActionUp = activateOnActionUp;
   }

   /**
    * creates a nea button with an  image
    */
   public Button(float posX, float posY, Bitmap image, boolean shouldBePressed, boolean
activateOnActionUp) {
       this.x = posX;
       this.y = posY;
       this.width = image.getWidth();
       this.height = image.getHeight();
       this.image = image;
       this.shouldBePressed = shouldBePressed;
       this.activateOnActionUp = activateOnActionUp;
   }

   /**
    * get the button's image
    */
   public Bitmap getImage() {
       return image;
   }

   /**
    * gets the button's x position
    */
```

```
public float getX() {
    return x;
}

/**
 * gets the button's y position
 */
public float getY() {
    return y;
}

/**
 * set the text inside the button
 */
public void setText(String text) {
    this.text = text;
}

/**
 * @return
 * true if the button should activate' false if not
 */
public boolean shouldActivate(){
    return activate;
}

/**
 * draws the button
 */
public void draw(Canvas canvas){
    if (image == null) {
        Paint paint = new Paint();
            paint.setARGB(255, 120, 120, 120);
            if (clicked)
                paint.setARGB(255, 100, 100, 100);
            canvas.drawRect(x, y, x + width, y + height, paint);
            paint.setARGB(255, 10, 10, 10);
            paint.setTextAlign(Paint.Align.CENTER);
            float textSize = getTextSizeForHeight(paint, height * 0.75f, "Testing");
            paint.setTextSize(textSize);
            int xPos = (int) (x + width / 2);
            int yPos = (int) (y + height / 2 - (paint.descent() + paint.ascent()) / 2);
            canvas.drawText(text, xPos, yPos, paint);
    }
    else {
        Paint tpaint =new Paint();    //original idea™
        if (clicked){
            tpaint.setAlpha(150);}
        canvas.drawBitmap(image, x, y, tpaint);
    }
}

/**
 * updates the buttons state, active or not
 */
public void update(){
    activate = false;
```

```
        if (activateOnActionUp) {
            if (clicked && Input.upPos != null) {
                float x = Input.upPos.x;
                float y = Input.upPos.y;
                if (x >= this.x && x <= this.x + width && y >= this.y && y <= this.y + height) {
                    activate = true;
                }
            }
        }

        boolean newClicked = false;
        SparseArray<PointF> touchPoints = Input.getTouchPoints();
        if(touchPoints.size() == 0)
            pressed = false;
        for (int i = 0; i < touchPoints.size(); i++) {
            int key = touchPoints.keyAt(i);
            if(shouldBePressed && !Input.down && !pressed)
                break;
            if(shouldBePressed && Input.downId != key && !pressed) {
                continue;
            }
            if(touchPoints.get(key) == null) {
                continue;
            }
            float x = touchPoints.get(key).x;
            float y = touchPoints.get(key).y;
            if (x >= this.x && x <= this.x + width && y >= this.y && y <= this.y + height) {
                newClicked = true;
                pressed = true;
            }
        }
        if(!newClicked && pressed)
            pressed = false;
        clicked = newClicked;
        if (!activateOnActionUp)
            activate = clicked;
    }
    //util
    /**
     * @return
     * returns the text size to fit inside the button
     */
    private float getTextSizeForHeight(Paint paint, float desiredHeight, String text) {
        // Pick a reasonably large value for the test. Larger values produce
        // more accurate results, but may cause problems with hardware
        // acceleration. But there are workarounds for that, too; refer to
        // http://stackoverflow.com/questions/6253528/font-size-too-large-to-fit-in-cache
        final float testTextSize = 48f;
        // Get the bounds of the text, using our testTextSize.
        paint.setTextSize(testTextSize);
        Rect bounds = new Rect();
        paint.getTextBounds(text, 0, text.length(), bounds);

        // Calculate the desired size as a proportion of our testTextSize.
        float desiredTextSize = testTextSize * desiredHeight / bounds.height();
        // Set the paint for that size.
        return desiredTextSize;}}
```

Pref

```
package com.arg.hmmm.carhell;

import android.content.Context;
import android.content.SharedPreferences;
import android.view.View;

public class Pref {
    private static SharedPreferences pref;

    /**
     * the Pref class is used to store game options after the closing the game
     */
    public Pref() {
    }

    /**
     * creates a new shared preferance
     */
    public static void createPref(Context context) {
        pref = context.getSharedPreferences("myPrefsKey", Context.MODE_PRIVATE);
    }

    /**
     * @return
     * corrent highscore
     */
    public static float getHighScore() {
        return pref.getFloat("score", 0);
    }

    /**
     * sets a high score
     */
    public static void setHighScore(float score) {
        SharedPreferences.Editor edit = pref.edit();
        edit.putFloat("score", score);
        edit.commit();
    }

    /**
     * sets is the big buttons option is enabled or not
     */
    public static void setBigButtons(boolean b) {
        SharedPreferences.Editor edit = pref.edit();
        edit.putBoolean("bigButtons", b);
        edit.commit();
    }

    /**
     * @return
     * true if the big buttons option is enabled and false if not
     */
    public static boolean getBigButtons() {
        return pref.getBoolean("bigButtons", false);
    }
```

```java
    /**
     * sets the player's corrent image
     * the player's image can be chosen it it saves after closing the game
     */
    public static void setPlayerImage(String s) {
        SharedPreferences.Editor edit = pref.edit();
        edit.putString("playerImage", s);
        edit.commit();
    }

    /**
     * @return
     * the last set player image
     */
    public static String getPlayerImage() {
        return pref.getString("playerImage", "yellow");
    }

    /**
     * sets is the volume on or off
     */
    public static void SetVolumeState(boolean b){
        SharedPreferences.Editor edit = pref.edit();
        edit.putBoolean("volumeState" , b);
        edit.commit();
    }

    /**
     * @return
     * true if the volume is active and false if volume is turned off
     */
    public static boolean getVolumeState(){
        return pref.getBoolean("volumeState", true);
    }

}
```

Thing

```
package com.arg.hmmm.carhell;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Matrix;
import android.graphics.Paint;

public class Thing implements Runnable{
   protected GameView view;
   protected Bitmap bmp;
   protected Matrix matrix;
   protected Paint paint;

   protected float x;
   protected float y;
   protected float vx;
   protected float vy;

   protected Thread thread;
   protected boolean needToStop;

   /**
    * creates a new Thing  at (x,y) without an image
    * @param x
    * thing's x value
    * @param y
    * thing's y value
    */
   public Thing(float x, float y, GameView view){
      this.x=x;
      this.y=y;
      this.view=view;
      needToStop = false;

   }

   /**
    * creates a new thing at (x,y)  with a provided image
    * @param x
    * thing's x value
    * @param y
    * thing's y value
    * @param vx
    * thing's horisontal speed (on x axis)
    * @param vy
    * thing's vertical speed (on y axis)
    * @param bmp
    * thing's image
    */
   public Thing(float x, float y, GameView view, float vx, float vy, Bitmap bmp){
      this(x,y,view);
      this.vx=vx;
      this.vy=vy;
      this.bmp=bmp;
```

```
        thread =new Thread(this,"ThreadNum1");
        thread.start();
    }

    //getters
    /**
     * returns the thing's x value
     */
    public float getX(){
        return this.x;
    }
    /**
     * returns the thing's y value
     */
    public float getY(){
        return this.y;
    }
    /**
     * returns the thing's horisontal speed (on the x axis)
     */
    public float getVx(){
        return this.vx;
    }
    /**
     * returns the thing's vertical speed (on the y axis)
     */
    public float getVy(){
        return this.vy;
    }
    /**
     * return the thing's image width
     */
    public float getBmpWidth() {
        return bmp.getWidth();
    }
    /**
     * return the thing's image height
     */
    public float getBmpHeight() {
        return bmp.getHeight();
    }
    /**
     * returns the thing's image
     */
    public Bitmap getBmp() {
        return bmp;
    }

    //setters
    /**
     * sets the thing's x value
     */
    public void setX(float x){
        this.x=x;
    }
    /**
```

```
  * sets the thing's y value
  */
public void setY(float y){
    this.y=y;
}
/**
  * set the thing's horizontal speed (the the x axis)
  */
public void setVx(float vx){
    this.vx=vx;
}
/**
  * set the thing's vertical speed (on the y axis)
  */
public void setVy(float vy){
    this.vy=vy;
}
/**
  * set the thing's image
  */
public void setBmp(Bitmap bmp){
    this.bmp=bmp;
}
/**
  * stops the thing's thread
  */
public void stop(){
    needToStop=true;
}
/**
  * start the thing's  thread
  */
public void start(){
    needToStop=false;
    Thread thread = new Thread(this);
    thread.start();
}

/**
  * adds to the player's horizontal speed (on the x axis)
  */
public void addVx(float x){
    vx+=x;
}

/**
  * adds to the player's vertical speed (on the y axis)
  */
public void addVy(float x){
    vy+=x;
}


/**
  * draws the thing at (x,y)
  */
public void draw(Canvas canvas){
        //canvas.drawBitmap(bmp,x,y,paint);
        canvas.setMatrix(matrix);
        canvas.drawBitmap(bmp, x, y, paint);
        canvas.setMatrix(null);
    }
    /*public void drawMatrix(Canvas canvas){
        canvas.setMatrix(matrix);
        canvas.drawBitmap(bmp, x, y, paint);
        canvas.setMatrix(null);
    }*/

    /**
     * checks if the thing is outside the phone/game's borders, if so deletes it using remove()
     */
    private void checkRemove(){
        if(x<-bmp.getWidth()*2 || y< -bmp.getHeight()*2 || x>view.getScreenWidth()+bmp.getWidth() ||
y> view.getScreenHeight()+bmp.getWidth()){
            remove();
        }
    }

    /**
     * deletes this thing
     */
    private void remove(){
        needToStop = true;
        view.removeThing(this);
    }

    /**
     * moves  the thing according to its speed on the x and y axes (vx and vy)
     */
    @Override
    public void run() {
        while(!no()){
            checkRemove();

            x+=vx;
            y+=vy;

            try {
                Thread.sleep(15);
            } catch (Exception e){
                e.printStackTrace();
            }
        }
    }

    /**
     * returns true if the player needs to stop and false if he needs to continue
     * used inside run()
     */
    private boolean no(){
        return needToStop;
    }
}
```

Player

```java
package com.arg.hmmm.carhell;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Matrix;

import java.util.ArrayList;
import java.util.Timer;
import java.util.TimerTask;

public class Player extends Thing{
    private boolean isAccelerating;
    private float speed;
    private float angle;
    private float movementFixer;
    private final float speedFactor = view.getScreenHeight()/200f;
    private boolean isSheildActive;
    private TimerTask powerUpDuration;
    private Bitmap shieldBmp;

    /**
     * Creates a new player
     *
     * @param x
     * the player's x value
     * @param y
     * the player's y value
     * @param bmp
     * the image of the player
     */
    public Player(int x, int y, GameView gameView, Bitmap bmp){
        super(x,y,gameView);
        vy=0;
        vx=0;
        this.bmp = bmp;
        matrix = new Matrix();

        shieldBmp =
Bitmap.createScaledBitmap(BitmapFactory.decodeResource(view.getResources(),R.drawable.shield),
view.getScreenWidth()/12, view.getScreenWidth()/12, false);

        isAccelerating=false;
        //-90 to point upwords
        angle = 0;
        speed = 0;

        isSheildActive = false;

        movementFixer = (float)Math.toRadians(-90);


        thread =new Thread(this,"ThreadNum1");
        thread.start();
```

```
}

/**
 * sets is the player accelerating or not
 * @param x
 * true or false
 */
public void setIsAccelerating(boolean x){
    isAccelerating=x;
}

/**
 * sets the player's speed
 */
public void setSpeed(float x){
    this.speed=x;
}

/**
 * set the angle in wich the player moves
 */
public void setAngle(float x){
    this.angle = x;
}

/**
 * turns the player right
 */
public void turnRight(){
    //speed !=0
    if(Math.abs(speed) >= 1){
        angle += (float)Math.toRadians(5);
        rotateImage();
    }

}

/**
 * turns the player left
 */
public void turnLeft(){
    //speed !=0
    if(Math.abs(speed) >= 1){
        angle -= (float)Math.toRadians(5);
        rotateImage();
    }

}

/**
 * reduses the playes speed faster then normal
 * if the speed drops to 0 the player will start accelerating backwords
 */
public void brake(){
    if(speed-speedFactor*1.5 >= 0)
        speed -=2; // speedFactor*1.5
        else if(speed <= 0)
            speed -= 1.5; //speedFactor
    }

    /**
     * rotats the player's image depending on the player's angle
     */
    public void rotateImage(){
        /*
        matrix.reset();
        matrix.setTranslate(x, y);
        matrix.postRotate(angle, x+getBmpWidth()/2,y+getBmpHeight()/2);*/
        /*
        matrix.reset();
        matrix.setTranslate( x, y );
        matrix.postRotate(angle , x,y);
        matrix.postTranslate(x, y);
        bmp = Bitmap.createBitmap(bmp, 0, 0, (int)getBmpWidth(), (int)getBmpHeight(), matrix, true);*/

            matrix.reset();
            matrix.setRotate((float) Math.toDegrees(angle), x + bmp.getWidth() / 2f, y + bmp.getHeight() /
2f);

            /*
            -(float)Math.toRadians(-90)
            in order of the car to pint up the starting ange needs to be -90(rad)
             */
    }

    /*    //this function has been moved to Thing
    public void drawMatrix(Canvas canvas){
        canvas.setMatrix(matrix);
        canvas.drawBitmap(bmp, x, y, null);
        canvas.setMatrix(null);
    }*/


    /**
     * checks if the player is touching the phone/game boundaries
     */
    private void checkBoundaries(){
        if(y+getBmpHeight()/2<0) {
            y = view.getScreenHeight()-getBmpHeight()/2;
        }else if(y+getBmpHeight()/2>view.getScreenHeight()){
            y=0;}

        if(x+getBmpWidth()/2<0) {
            x = view.getScreenWidth()-getBmpWidth()/2;
        }else if(x+getBmpWidth()/2>view.getScreenWidth()){
            x=vx;}
    }

    /**
     * check for impact between the player and a thing
     */
    private void checkImpact(){
        ArrayList<Thing> things = view.getThings();
        if (things != null) {
            Thing tThing;
            float ty = y+(getBmpHeight()-getBmpWidth())/2;

            for(int i=0; i<things.size();i++){
                tThing = things.get(i);
                if(x+bmp.getWidth()>=tThing.getX()&& tThing.getX()+tThing.getBmpWidth()>=x &&
                        ty+bmp.getHeight()>=tThing.getY()&& tThing.getY()+tThing.getBmpHeight()>=ty){
                    if(tThing instanceof PowerUp){
                        switch(((PowerUp) tThing).getType()){
                            case "shield":
                                activateShild();
                                view.removeThing(tThing);
                                break;
                            default:
                                view.removeThing(tThing);
                        }
                    }
                    else
                        view.looseScrean();
                }

            }
        }
    }

    /**
     * activates the shield power-up making the player invulnerable for a couple of seconds
     */
    private void activateShild(){
        isSheildActive = true;
        Timer powerUpDorationTimer = new Timer();
        powerUpDuration = new TimerTask() {
            @Override
            public void run() {
                //add here something to stop, if()
                isSheildActive = false;
            }
        };
        powerUpDorationTimer.schedule(powerUpDuration,1000*5);
    }

    /**
     * draws the player and the shield (if active)
     */
    @Override
    public void draw(Canvas canvas){
        //canvas.drawBitmap(bmp,x,y,paint);
        super.draw(canvas);
        if(isSheildActive){
            canvas.setMatrix(matrix);
            canvas.drawBitmap(shieldBmp, (x + bmp.getWidth() / 2f)-shieldBmp.getWidth()/2f, (y +
bmp.getHeight() / 2f)-shieldBmp.getHeight()/2f, paint);
            canvas.setMatrix(null);
        }
    }
    /**
     * moves the player player depending on his speed, angle and is he accelerating
     */
    @Override
    public void run() {
        while(yes()) {
            x += vx;
            y += vy;

            view.getTestPixle().setX(x);
            view.getTestPixle().setY(y+(getBmpHeight()-getBmpWidth())/2);

            rotateImage();

            if (!isAccelerating) {
                if(speed-1>=0) //speed-speedFactor/2
                    speed -= 1;// speedFactor/2
                else if(speed+1<=0)//speed+speedFactor/2
                    speed+=1;// speedFactor/2
                else
                    speed = 0;
            } else {
                if(speed<speedFactor*4)//speedFactor*4
                    speed +=2;//speedFactor
            }
            checkBoundaries();

            if(!isSheildActive)
                checkImpact();

            //movementFixer is recuiered else when the car points upwords it will move to the right
            vx = (float)(speed*Math.cos(angle+movementFixer));
            vy = (float)(speed*Math.sin(angle+movementFixer));

            try {
                Thread.sleep(15);
            } catch (Exception e){
                e.printStackTrace();
            }
        }
    }

    /**
     * returns true if the player need to continue and false if he needs to stop
     * used inside run()
     */
    private boolean yes(){
        return !needToStop;
    }
}
```

PowerUp

```
package com.arg.hmmm.carhell;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.graphics.Paint;


import java.util.Timer;
import java.util.TimerTask;

public class PowerUp extends Thing {
    private TimerTask secondsTask;
    private String type;

    public String getType() {
        return type;
    }

    /**
     * creates a new power-up
     */
    public PowerUp(float x, float y, String type, GameView gameView){
        super(x,y,gameView);
        this.type = type;
        matrix = new Matrix();
        paint = new Paint();

        // made it so it would be easier to add more powerups in the future
        switch(type){
            case "shield":
                bmp = 
Bitmap.createScaledBitmap(BitmapFactory.decodeResource(gameView.getResources(),R.drawable.sh
ieldcolor), view.getScreenHeight()/12, view.getScreenHeight()/12, false);
                break;
            default:
                bmp = 
Bitmap.createScaledBitmap(BitmapFactory.decodeResource(gameView.getResources(),R.drawable.w
hitepixle), view.getScreenHeight()/12, view.getScreenHeight()/12, false);
        }

        lifeTimer();

    }

    /**
     * this functions has a timer that counts how long the until the PowerUp disappears
     * the PowerUp only appears for a certain length of time
     */
    private  void lifeTimer(){
        Timer secondsTimer = new Timer();
        bmp.setHasAlpha(true);
        secondsTask = new TimerTask() {
            int alfa = 255;
            int sycle = 0;
        boolean b = true;
        @Override
        public void run() {
            //add here something to stop, if()
            if(b){
                if(alfa >90)
                    alfa -= 5;
                else
                    b = false;
            }
            else{
                if(alfa < 255)
                    alfa += 5;
                else
                    b = true;
            }

            paint.setAlpha(alfa);
            sycle++;
            if(sycle == 500)
                removePowerUp();
        }
    };
    secondsTimer.scheduleAtFixedRate(secondsTask, 1000 * 5,10);
  }

  /**
   * removes the shield
   */
  private void removePowerUp(){
    view.removeThing(this);//if executed inside timer this = timer
  }
}
```

סיכום אישי

העבודה על הפרויקט הייתה עבודה מהנה ומעשירה, זו הייתה הפעם הראשונה שבה יצרתי תוכנה למכשיר טלפון (אנדרויד) והמשחק הראשון בסגנון הזה שייצרתי.

העבודה על הפרויקט לא הייתה תמיד קלה, ישנם עוד דברים רבים שרציתי להוסיף למשחק, כגון: עוד מכוניות ושיהיה הבדל ביניהם חוץ מהמראה שלהם (מכונית יותר מהיר/משאית שהיא יותר איטית וכדומה, הישגים למיניהם (achivments), עוד powerups וכו'. אבל למרות הקשיים והמחסור בזמן, במיוחד לאור הנסיבות האחרונות במדינה, אני שמח ומרוצה מהפרויקט שייצרתי.

למדתי רבות בזמן העבודה על הפרויקט ואני בטוח שמה שיצרתי הוא משחק קטן וכיף שכל אחד יכול לשחק בו ולהנות ממנו.

ביבליוגרפיה

במהלך העבודה המקור העיקרי שהשתמשתי בו הוא האתר StackOverflow

https://stackoverflow.com/