הקדמה:

למה בחרנו לפתח אפליקציות לגוגל אנדרואיד ולא לאייפון?

ישנו מספר רב של סיבות מדוע לפתח אפליקציות למערכת ההפעלה אנדרואיד ולאו דווקא לאייפון. אנו בטוחים שיש הרבה מפתחים אשר מתלבטים לאיזה מערכת הפעלה לפתח אפליקציות (apps), ההתלבטות הזו גם הייתה קיימת אצלנו עד לא מזמן. אנו משקיעים זמן רב בלימוד של שפת תכנות חדשה לסביבות פיתוח שונות, אם זה ASP.NET ושפת C# או לבניית אתרי אינטרנט PHP שפת

כמו רוב המפתחים גם לנו הייתה התלבטות כאשר היינו צריכים לבחור לאיזה פלטפורמה סלולרית לפתח אפליקציות. מצד אחד האייפון מאוד קרץ לנו: קהל יעד גדול, פיתוח למכשיר אחד עם רכיבי חומרה ידועים וגרסה עדכנית אחת למכשירים החדשים, חנות אפליקציות ענקית (App Store) ועוד. לעומת זאת, כאשר גוגל הכריזה על מערכת ההפעלה החדשה Google Android OS, שלה היה הד תקשורתי אדיר מיד הבנו שמדובר במשהו חדש חזק מאוד בתוך השוק של מערכות ההפעלה Google ומהפכני, אשר יתכן שיציב את חברת לטלפונים סלולריים.

הסיבות שבחרנו דווקא לפתח אפליקציות למערכת הפעלה אנדרואיד של גוגל הן כדלקמן:

תכנות בשפת JAVA - מפני שהיא יכולה לשמש אותנו JAVA העדפנו לפתח בשפת לפיתוח אפליקציות אחרות לסלולר לטלפונים אחרים. השפה עצמה קלה ללמידה למי זוהי השפה המועדפת על הרבה מתכנתים אשר מפתחים C#. שעובר מתכנות בשפת אפליקציות למכשירים סלולריים.

מערכת הפעלה של Google - אין ספק שחלק גדול מההחלטה שלנו הוא הסיבה עומדת מאחורי פיתוח הפלטפורמה הנ"ל. אנו מאמינים כי גוגל Google שחברת ותציג טכנולוגיות Mobile-תשכיל לייצב את עצמה אל מול המתחרות שלה בתחום ה חדשות בתדירות גבוהה, ואלו יספקו אתגר למידה. בנוסף אנו מאמינים כי גוגל תשקיע API משאבים בלפתח אפליקציות 'ענן', אשר האינטגרציה שלהם בעזרת פונקציות למכשירים הסלולריים העובדים עם (Native) למכשירי אנדרואיד, תהיה טבעית מערכת ההפעלה שלה.

קוד פתוח - עוד סיבה טובה מדוע לבחור לפתח apps תחת Android : היתרון בקוד פתוח הוא החופש והיכולת להציץ בתוך קוד המקור, להפיץ גרסאות שונות של מערכת

מתי שמתבקש (OTA) ההפעלה (לא תמיד יתרון), לעדכן את מערכת ההפעלה מרחוק אם זה עדכון אבטחה או עדכון אחר למערכת ההפעלה לפתיחה של תוספים) עדכון כמו ה-Multi-Touch).

אפליקציות פתוחות - אין ספק הרבה מפתחים היו רוצים יותר חופשיות בהגשת המדיניות של אפל (App Store). האפליקציה שלהם לחנות האפליקציות של אפל קשוחה בהשוואה לזו של חנות האפליקציות של גוגל, שם אפשר להעלות אפליקציה מבלי לעבור סינון. מאידך, אילו היה סינון ייתכן שהיו פחות אפליקציות "זבל". מי שיוצר את הסינון הם המשתמשים, אשר מגיבים על האפליקציות, ובמידה ואפליקציה מסוימת נחשבת לזדונית, ספאמית או לא ראויה, היא לאחר מכן תוסר על ידי גוגל מחנות האפליקציות. אך מי שקובע זאת אלו הם הגולשים מראש ולא גוגל. כמו שכתבנו למעלה, זה יכול להיות חסרון מסוים כי אז הרבה אפליקציות 'זבל', לא איכותיות, יופיעו בהמוניהן בחנות האפליקציות של גוגל Google Play.

סביבת פיתוח Eclipse IDE - סביבת הפיתוח הפופולרית בעולם לפיתוח יישומי JAVA הנקראת Eclipse. פלטפורמה בוגרת ומתקדמת, דומה מאוד לזו של מיקרוסופט Visual Studio. כזה, הופך את IDE אין ספק שפיתוח אפליקציות עם הפיתוח לחוויה. בנוסף לאחר זמן קצר לאחר ההשקה של מערכת ההפעלה החדשה, סביבת פיתוח MOTODev. הנקראת Eclipse-חברת מוטורולה הוציאה גרסה שלה ל רק כאשר בהתקנה אחת אפשר לקבל את כל Eclipse, של IDE-המבוססת על ה הכלים שצריך בשביל לפתח אפליקציות אנדרואיד, כולל אמולטור מובנה בתוך סביבת הפיתוח ולא חיצוני.

קהל יעד - אנו מאמינים כי מערכת ההפעלה Android OS תהיה מערכת ההפעלה הפופולרית ביותר בעוד כמה שנים. נכון שעכשיו עדיין אפל מוכרת יותר מכשירים מאשר מכשירים מבוססי אנדרואיד. אך אין ספק שבהתחשב בזמן שהפלטפורמה עצמה קיימת, היא מגדילה את הפופולריות שלה בקצב מסחרר.

מכשירים סלולריים - אין ספק שתמיד יש מקום להתלהבות כאשר יוצא מכשיר סלולרי חדש, לא משנה אם זה מכשיר עם פלטפורמת Google Android או פלטפורמה אחרת. אך אם זאת, נכון לסוף שנת 2012 שוק הסלולרים מורכב ב-46% מכשירים שמריצים את מערכת ההפעלה Google Android, ותמיד מעניין לראות את אותם המכשירים החדשים שבשבילם אנו נפתח אפליקציות חדשות. כמובן זה גם יביא ליותר חשיפה למערכת ההפעלה החדשה, יותר אופציות של אנשים אשר אין להם את הכסף להשקיע במכשיר אייפון חדש, לקנות דווקא טלפונים חדשים של חברות LG למשל, אשר השיקו בשנת 2012 מיליוני מכשירים סלולריים חדשים מבוססי HTC-ו Android, אכן מגניב!

היינו חייבים להחליט - אין ספק שפשוט צריך להחליט בסופו של דבר, וכמו שאפשר לראות, היו לנו מספיק סיבות מדוע דווקא לבחור במסלול הזה. אין ספק שיש חסרונות לפיתוח אפליקציות למספר גדול של מכשירים ומערכות הפעלה שונות, ואין ספק

בעולם בכלל ובישראל בפרט. בכל -iPhone שקשה להתעלם מהפופולריות של ה זאת, צריך פה קצת אמונה במוצר (בדיוק כמו במניה). אנו מאמינים במוצר שגוגל מוכרת!

מבוא ל***JAVA:***

ג'אווה (Java) היא שפת תכנות מונחית עצמים אשר פותחה בחברת סאן מיקרוסיסטמס (כיום חברת-בת של אורקל) על ידי צוות בראשות ג'יימס גוסלינג בשנת 1991, והיא אחת משפות התכנות הנפוצות ביותר הנמצאות בשימוש כיום. השפה שוחררה לראשונה בשנת 1995, והיא מהווה את אחד מרכיבי הליבה של פלטפורמת התוכנה ג'אווה.

אך כולל הרחבות ++C, התחביר של השפה מבוסס במידה רבה על התחביר של רבות במטרה לאפשר תמיכה מובנית בתהליכונים, בינלאומיות, אבטחה ועבודה בסביבת האינטרנט, ותכונות נוספות. לרוב עוברות תוכניות ג'אווה הידור ל-Java bytecode, שפת ביניים דמוית שפת מכונה, שאותה מריצה מכונה וירטואלית (Java Virtual Machine; JVM). הודות לכך התוכנית יכולה לרוץ על כל מחשב ועל כל מערכת הפעלה המריצים JVM, החל מטלפונים סלולריים ועד למחשבי על.

ג'אווה היא שפה בעלת טיפוסיות סטטית חזקה, כלומר לכל ביטוי בשפה מתאים טיפוס יחיד, תקינות ביטויים נבדקת בזמן ההידור, במידת האפשר. כאשר אין זה אפשרי, תתבצע בדיקה בזמן ריצה.

ג'אווה תומכת בתכנות מונחה עצמים. עם זאת ג'אווה עדיין כוללת, מטעמי יעילות, גם "טיפוסים פרימיטיביים" (במקור primitive types), שאינם אובייקטים, ועל כן אינה נחשבת שפה מונחית עצמים "טהורה" כמו Smalltalk או Ruby. עם זאת, לכל טיפוס פרימיטיבי בג'אווה יש מחלקה עוטפת, אליה וממנה מתבצעת המרה אוטומטית (Boxing). למשל, לטיפוס הפרימיטיבי int קיימת מחלקה עוטפת Integer.

שפת ג'אווה כוללת גם ניהול זיכרון אוטומטי. המתכנת פטור מן ההכרח לשחרר זיכרון המוקצה לאובייקט ברגע שאין עוד משתנים המצביעים עליו. במקום זאת, סביבת זמן הריצה מבצעת "איסוף זבל", המבצע זאת אוטומטית.

ג'אווה מאפשרת ירושה יחידה בלבד, וזאת על מנת למנוע בעיות ++C, בניגוד לשפת דו-משמעות הנוצרות מירושת יהלום. כדי לאפשר גמישות דומה לזו של ירושה מרובה, המגדיר רשימה של מתודות, המהווה חוזה עם (interface) קיים בג'אווה מנגנון ממשק לכל היותר מחלקה אחת, (extends) המתכנת. כך כל מחלקה יכולה להרחיב מספר בלתי מוגבל של ממשקים. על המחלקה לממש (implements) ולממש במפורש כל מתודה בכל ממשק כזה, וכך לא קיימת בעיה של כפל משמעות.

בגרסה 5.0 נוספו מספר הרחבות חשובות. הגרסה הנוכחית של ג'אווה היא JAVA SE 7. מספר משתנה של פרמטרים לפונקציה וטיפוסי (Generics), לשפה, ביניהן תכנות גנרי מניה (Enums).

מבוא לאנדרואיד:

אנדרואיד - מערכת הפעלה למכשירים ניידים. קצת היסטוריה: מערכת ההפעלה אנדרואיד נרכשה מחברת אנדרואיד על ידי גוגל בשנת 2005. בנובמבר 2007 הוכרז שמטרתה פיתוח, (Open Handset Alliance- OHA) על הקמתה של ברית חברות OHA-סטנדרטים פתוחים למכשירים ניידים. הפיתוח הראשון עליו הכריזה ה היתה מערכת ההפעלה אנדרואיד המבוססת על לינוקס קרנל גרסה 2.6. בפברואר 2009 הופצה הגרסה הראשונה לשימוש כללי של לקוחות - גרסה 1.1.

אז מהי אנדרואיד בעצם?

אנדרואיד הינה מערכת תוכנה הבנויה בשכבות ומיועדת למכשירים ניידים כגון טלפונים וטבלטים. מערכת

- זוהי מערכת הפעלה המיועדת למכשירים ניידים.
- מבוססת על גרעין לינוקס.
- שפותחה (JVM - Java Virtual Machine) במרכזה מכונת ג'אווה וירטואלית המשמשת למערכת האנדרואיד הוא דרך Dalvik. במיוחד למערכת ושמה דלוויק ספריות ג'אווה.
    - הערה: קיימת למעשה אפשרות להתממשק למערכת גם תוך בשפת NDK- Native Development Kit ושימוש ב JVM-מעקף ה C/C++.

אנדרואיד כוללת מערכת הפעלה, תשתית לבניית אפליקציות, מכונה וירטואלית להרצת האפליקציות, ספריות גרפיות, בסיס נתונים, תמיכה במדיה, טלפוניה, רשת חיישנים וכן סביבת פיתוח, GPS, אלחוטית, מצלמה, ניווט עשירה המשתלבת במערכת הפיתוח Eclipse.
אנדרואיד כוללת גרעין של מערכת ההפעלה לינוקס. היא מגיעה עם קבוצת אפליקציות בסיסיות הכוללות:
לוח שנה, מפות, דפדפן, אנשי קשר ועוד. SMS, אפליקציה לדואר אלקטרוני
בשפת כתובות אחרות ורבות , Javaאפליקציות רבות כתובות בשפת Native C programming language)).
סביבת הפיתוח הפתוחה מאפשרת למפתחים לבנות אפליקציות חדשניות ועשירות, עם גישה מלאה לחומרת המכשיר, למערכת המיקום, התראות והודעות, ניתן להריץ

ברקע ועוד הרבה אופציות אחרות. כל זאת תוך שמירה על (services) שירותים אבטחת המידע של המשתמש, ולכן כל גישה כזאת דורשת את אישורו (בעת התקנת האפליקציה) בכתיבת אפליקציה ניתן להשתמש בסט של אלמנטים גרפיים הנקראים widgets או views (לדוגמא: lists, grids, text boxes, buttons ועוד).
ניתן לגשת למידע של אפליקציות אחרות בעזרת ספקי תוכן, או לשתף את המידע של האפליקציה שכותבים.

אבני הבניין של אפליקציה:

Activity	**שמתאים *UI* הרכיב הכי בסיסי, בד"כ למסך אחד שלם**
IntentReceiver	**מגיב להודעות \ שינויי סטאטוס, יכול להעיר תהליך כלשהו**
Service	**שרצה ברקע *UI* משימה ללא**
ContentProvider	**מאפשר לאפליקציה לשתף מידע עם אפליקציות אחרות**

מעגל החיים של אפליקציה:

כל אפליקציה רצה בתוך תהליך משלה

Security, protective memory

אפליקציה שמעמיסה על המעבד לא תחסום תהליכים קריטיים אחרים כגון לענות לטלפון

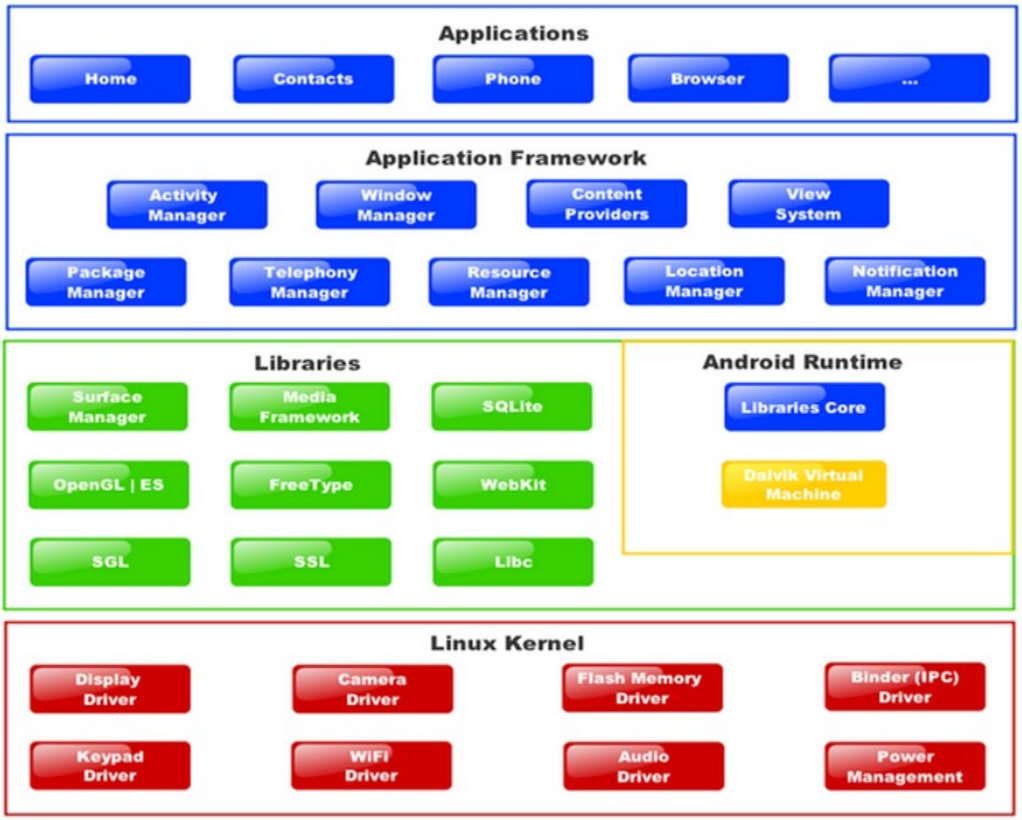
אחראית להפעלתם ועצירתם של תהליכים כנדרש להרצתן של Android מערכת אפליקציות

יתכן שתהליכים יהרגו (ואח"כ ישוחזרו) על מנת לפנות משאבים לתהליכים אחרים

מבנה חבילת התוכנה של האנדרואיד כולל מספר שכבות:

שכבת האפליקציה הרצה על מסגרת של ג'אווה בשיטת תכנות מונחה עצמים, מתחתיה ספריות ג'אווה הרצות על גבי מכונת ג'אווה וירטואלית בשם דלוויק (על שם הכפר, הנמצא באיסלנד כמדומני, ממנו באו אבותיו של מפתח המכונה).

נמצא גרעין הלינוקס והדרייברים, C מתחת לספריות הכתובות בשפת.
המבנה הארכיטקטוני של שכבות התוכנה מתואר בדיאגרמת הבלוקים הבאה.



מספר מושגי יסוד של האנדרואיד:

גם אם בקריאה ראשונה ההגדרות עלולות להראות אבסטרקטיות מדי, אין ספק שבהמשך הן תתבהרנה.

1) Activity

2) Intent

3) Service

Activity - כל תצוגה על ממשק המשתמש (UI = User Interface) היא האפליקציה יכולה לכלול סדרה של תצוגות, שכל אחת מהן היא אקטיביטי Activity.

אחראית לשמור את הנתונים הקשורים אליה, כך שאם היא Activity בפני עצמה. כל תופסק ותופעל מחדש, התצוגה תשתחזר. הרחבה על היכולת לשחזר אינפורמציה בפרק שדן ב מחזור חיי האפליקציה.

Intent - זהו המנגנון באמצעותו מועברות הודעות במערכת. בעזרת מנגנון זה ניתן ליצור קשר עם אלמנטים בתוך האפליקציה ומחוצה לה, וכך לבצע פעולות כמו:

- הרצוי Activity להפעלת Intent מעבר בין מסכי התצוגה שנעשה ע"י שליחת;
- העברת נתונים בין Activities;
- גישה לשירותים של מערכת האנדרואיד, למשל שליחת אימייל;
- יצירת קשר עם שירותים אחרים (ראה Service כאן בהמשך);

גישה למודולים של גורם שלישי הקיימים על הפלטפורמה, העברת הודעות מפורש ומרומז - Intent במערכת. קיימים שני טיפוסים של Broadcast.
ב-Intent מפורש (Explicit) במקרה של השולח מציין במפורש את שם יעד ההודעה. Intent מרומז (Implicit) השולח לא מציין את יעד ההודעה, אבל לפי הפרמטרים ליעד המתאים. מנגנון זה מאפשר קוד Intent Filter-שבה, ההודעה מכוונת ע"י ה גמיש בו אין צורך לדעת מראש את זהות היעד של הודעה.

Service - פעילות שרצה ברקע, ללא אינטראקציה ישירה עם המשתמש, כלומר בלי שהוא כאמור בעל ממשק Activities-יכול לקבל פקודות מ ה-Service UI. מתאים לפעילויות רקע כגון נגן, טיימר ה-Service Intent. למשתמש באמצעות וכדומה.

1. תיאור המשחק

פרק זה מספק את השלבים להגדרה ולאופן פעולה של מערכת ע"י משתמש.

מבוא לאפליקציה:

האפליקציה שלנו עוסקת בשיתוף מספר פעולות בין ויזואליזציה של משחק ובין חישה אקסילומטרית, כלומר יכולת המכשיר לחוש את מיקומו ואת כיוונו ביחס לכדור הארץ.
הרעיון המרכזי הוא לבנות משחק בשליטה על ידי חישת accelerometer.

מטרת הפרויקט:

- לימוד פיתוח אפליקציות לאנדרואיד.
- לעומק, ולכתוב בה קודים, מחלקות ותוכניות **Java** להבין את שפת.
- ליצור ממשק לעיבוד מידע ולשליפת תוצאות ממאגרי נתונים.
- להכיר טכניקות שונות בתכנות וללמוד כיצד משתמשים בפלטפורמות בנייה ובדיקה הידועות בשוק.
- ולהכיר תבניות תכנות **Android** לעצב אפליקציות להתקנים ניידים מבוססי ל-**Java**.
- פיתוח אפליקציות מסחריות, ותפעול סביבת ה-**Android.**

מה מייחד את הפרויקט שלנו: הפרויקט הוא ידידותי למשתמש, והממשק נגיש ופשוט לשימוש. אין צורך בידע מיוחד. והאפליקציה תוכל לשמש אנשים בכל הגילאים.

Manifest Android:

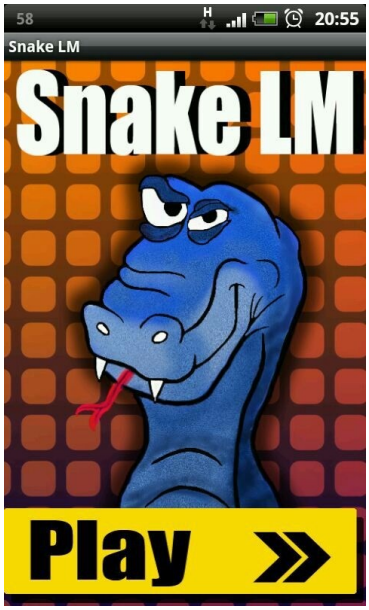
- המייצג את כל AndroidManifest.xml לכל אפליקציה חייב להיות קובץ בשם המידע החיוני על האפליקציה .כלפי מערכת האנדרואיד, לפני שהמערכת תוכל להריץ את הקוד.
  המשמש כמזהה ייחודי לאפליקציה, (package) הקובץ מציין את שם החבילה.
- בקובץ מוצהרים כל הרכיבי שמהם מורכבת האפליקציה: ה-broadcast activities ו- content provider וכן התכונות services, receivers, והיכולות שלהם.
  הוא מגדיר את ההרשאות שיש לתת למשתמשי האפליקציה, למשל גישה לאינטרנט, גישה לאנשי קשר, מצלמה וכו'. הוא מגדיר גם את גרסת האנדרואיד המינימלית שדרושה ואת הקונפיגורציה של החומרה שנדרשת. כמו כן את רשימת הספריות שיש ליצור קשר אליהן ועוד.
- **Strings.xml/values/res**: קובץ בו נמצאים כל המשאבים של התוכנה (מערכים של מחרוזות\צבעים\strings).
- **Attars/values/res**: קובץ Thems לתוכנה.
- **Layout/res**: בתיקיה זו נמצאים כל ה-xml (ממשקים משתמש) של כל

התוכנה, גם כל המסכים.

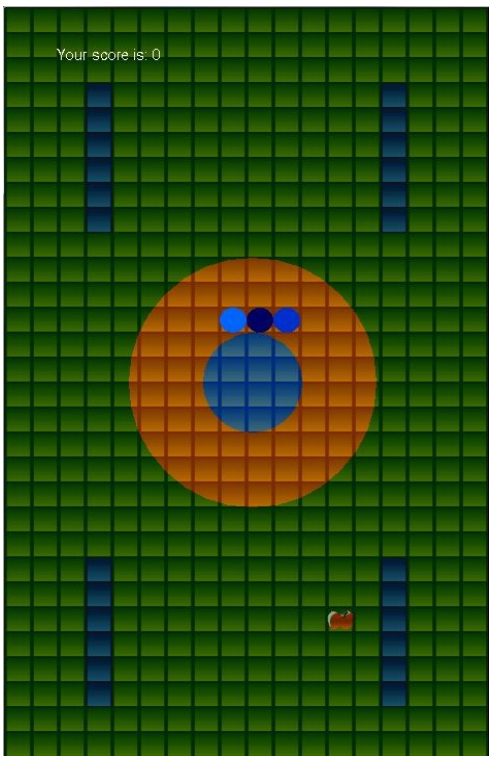
- **Drawble/res**: משאבים של תמונות. כל התמונות שבשימוש התוכנה נמצאים בתיקיות Drawble.
- **Anim/res**: קבצי xml של אפקטים בשימוש התוכנה. למשל הצגה של אובייקט כלשהו או מחרוזת באופן אפקטבי ומרשים.
- **Src** (**sourse**): ראשוני, קריאה של Activity-השם של חבילה. מתחילים מ **Manifest** היא ברירת מחדל לפי **Manifest**מתבצע ת **ב** Activity **SnakeLM Activity**.

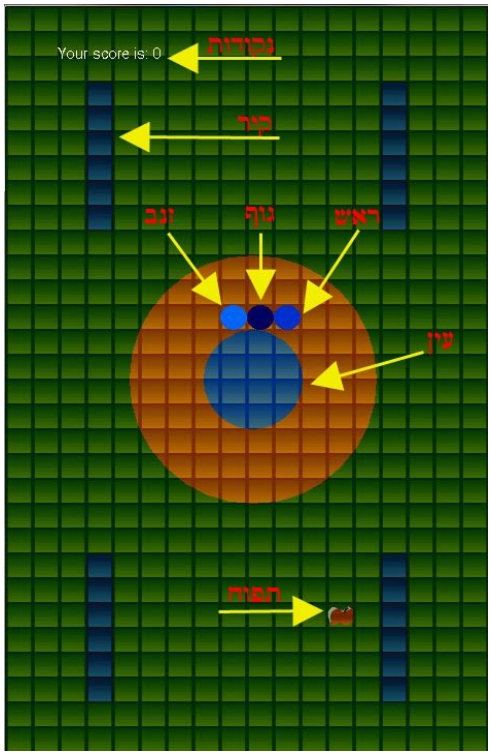
צילומי מסך של האפליקציה שלנו :

1. מסך כניסה לאפליקציה לאחר הפעלת האפליקציה על מסך מופיע תמונת רקע של משחק ואת הכפתור לתחילת משחק.



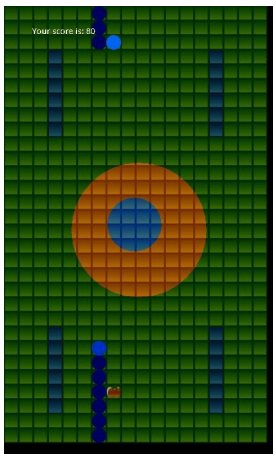
2. SNAKE מקבלים מצב התחלתי של "PLAY" אחרי לחיצה על



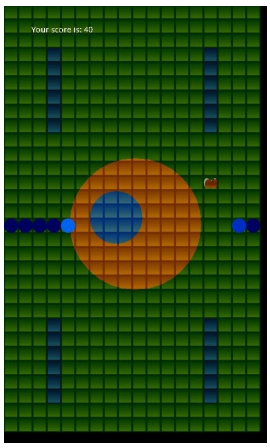


3. הנחש יכול לעבור דרך הקצה של המסכים שאנו רואים בתמונות הבאות.

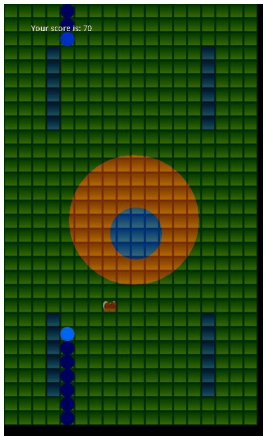
מעבר ממלמטה למעלה.



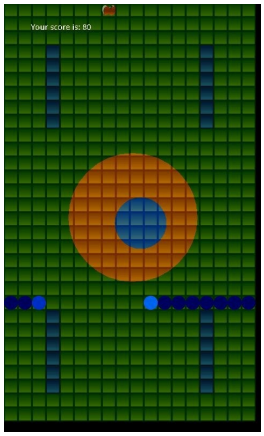
4. מעבר מימין לשמאל



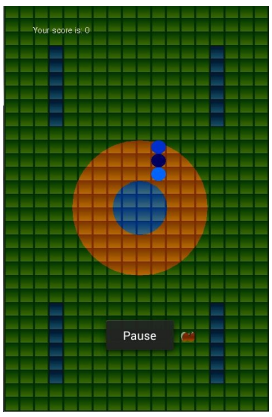
5. מעבר מלמעלה למטה.



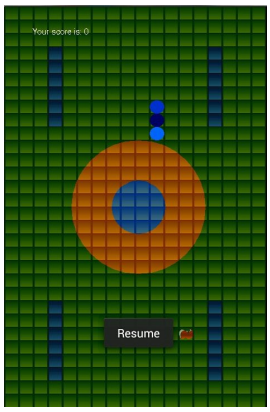
6. מעבר משמאל לימין



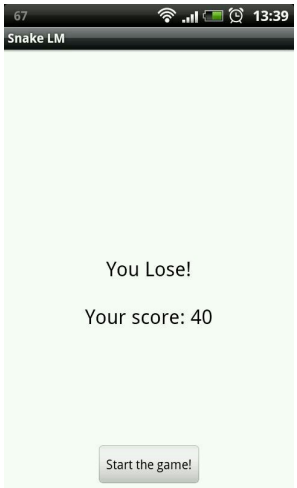
7. כשנוגעים במסך בזמן משחק פעם ראשונה, המשחק עוצר ומופיע כיתוב מתאים.



8. כשנוגעים במסך בזמן משחק פעם שניה, המשחק ממשיך בחזרה מהנקודה שעצרנו בה ומופיע כיתוב מתאים.



9. " מוות". אם נתקלנו באחד הקירות או בעצמנו, אז מקבלים מסך של סוף המשחק, את הנקודות שנאספו במהלך המשחק וכפתור כדי להתחיל לשחק מחדש שוב.



קוד המשחק SnakeGame

Class SnakeGame

```
// SnakeGame- מתאר את לוגיקה העומדת מאחורי המשחק
package com.snake.lm;
import java.util.ArrayList;

public class SnakeGame {

	// המחלקה מגדירה את המיקום
	public class pos {
		int x;
		int y;

		//constructor
		pos(int x, int y) {
			this.x = x;
			this.y = y;
		}
	}

	// כיוון
	public static final int DIR_NORTH = 1;
	public static final int DIR_EAST    = 2;
	public static final int DIR_SOUTH = 3;
	public static final int DIR_WEST   = 4;



	// רוחב וגובה של שדה המשחק
	public static int mFieldX = 18;
	public static int mFieldY = 30;
```

```
//חישוב נקודות במשחק
public int mScore=0;

// דו ממדית מטריצה
//שדה המשחק
private int mField[][] = new int[mFieldX][mFieldY];
// הגדרת נחש - מערך של קואורדינטות דו ממדי לכל מגזר
public ArrayList<pos> mSnake = new ArrayList<pos>();

//   כיוון נוכחי של הנחש
int mDirection = SnakeGame.DIR_EAST;

// constructor
SnakeGame() {
        // מנקים את השדה
        for (int i = 0; i < mFieldX; i++)
                for (int j = 0; j < mFieldY; j++) {
                        mField[i][j] = 0;
                }
        // יצירת הנחש
        mSnake.add(new pos(8, 10));

          //  לכל תא בתחום שבו
        //1 ציין - נחש נמצא-
        mField[8][10] = -1;
        mSnake.add(new pos(8, 11));
        mField[8][11] = -1;
        mSnake.add(new pos(8, 12));
```

```
mField[8][12] = -1;

// 1 =  הגדרת קיר
mField[3][3] = 1;
mField[3][4] = 1;
mField[3][5] = 1;
mField[3][6] = 1;
mField[3][7] = 1;
mField[3][8] = 1;

mField[14][3] = 1;
mField[14][4] = 1;
mField[14][5] = 1;
mField[14][6] = 1;
mField[14][7] = 1;
mField[14][8] = 1;

mField[3][27] = 1;
mField[3][26] = 1;
mField[3][25] = 1;
mField[3][24] = 1;
mField[3][23] = 1;
mField[3][22] = 1;

mField[14][27] = 1;
mField[14][26] = 1;
mField[14][25] = 1;
mField[14][24] = 1;
mField[14][23] = 1;
```

```
        mField[14][22] = 1;
        // מוסיפים פרי בשדה
        addFruite();
}

//המתודה המוסיפה את הפירות בשדה
// 2 - קוד של פרי בשדה

private void addFruite() {
        boolean par = false;
        while (!par) {
                int x = (int) (Math.random() * mFieldX);
                int y = (int) (Math.random() * mFieldY);
                if (mField[x][y] == 0) {
                        mField[x][y] = 2;
                        par = true;
                }
        }

}


        //המתודה זו מכילה את כל היגיון המשחק
        //כאן נתאר את כל הפעולות שיש לבצע בכל
        // תזוזת הנחש
        // ניקח בחשבון את הכיוון הנוכחי
        //נוסיף בדיקות האם הנחש יכול לזוז  בכיוון הזה

        public boolean nextMove() {
```

```
// בודקים לאיזה כיוון פונה נחש
switch (this.mDirection) {
// האם בצפון
case DIR_NORTH: {
// מחשבים את הקאורדינטות
//שיהיה ראש הנחש
//בצעד הבא
int nextX = mSnake.get(mSnake.size() - 1).x;
int nextY = mSnake.get(mSnake.size() - 1).y - 1;
// הצגת נחש על המסך
// ובדיקת האם תא הבאה פנויה לתזוזת הנחש
if (nextY >= 0){
//מקדמים גוף הנחש בשדה
if( mField[nextX][nextY] == 0) {
mField[mSnake.get(0).x][mSnake.get(0).y] = 0;
mSnake.remove(0);
// מקדמים ראש הנחש
mSnake.add(new pos(nextX, nextY));
mField[nextX][nextY] = -1;
return true;

// כאשר נתקענו בקיר מחזירים- שקר
} else if (mField[nextX][nextY] == 1) {
return false;
//כאשר נתקענו בפרי מוסיפים זנב לנחש
// מוסיפים נקודות הצלחה ופרי חדש לשדה
} else if (mField[nextX][nextY] > 1) {
mScore=mScore+10;
mField[nextX][nextY] = 0;
```

```
                mSnake.add(new pos(nextX, nextY));
                mField[nextX][nextY] = -1;
                addFruite();
                return true;
        }
}

//הצגת נחש מאחורי המסך
if ((nextY > mFieldY)||(nextY <0)) {
        nextY = mFieldY-1;

        if(mField[nextX][nextY] == 0){
                mField[mSnake.get(0).x][mSnake.get(0).y] = 0;
                mSnake.remove(0);
                mSnake.add(new pos(nextX, nextY));
                mField[nextX][nextY] = -1;
                return true;

        }else if(mField[nextX][nextY] == 1) {
                return false;

        }else if(mField[nextX][nextY] > 1) {
                mScore=mScore+10;
                mField[nextX][nextY] = 0;
                mSnake.add(new pos(nextX, nextY));
                mField[nextX][nextY] = -1;
                addFruite();
                return true;
        }
```

```
        }

        else {
                return false;
                }

}

// הגדרת כיוון מזרח דומה לשלב הקודם
case DIR_EAST: {
        int nextX = mSnake.get(mSnake.size() - 1).x + 1;
        int nextY = mSnake.get(mSnake.size() - 1).y;
        if (nextX < mFieldX) {
                if( mField[nextX][nextY] == 0) {
                        mField[mSnake.get(0).x][mSnake.get(0).y] = 0;
                        mSnake.remove(0);
                        mSnake.add(new pos(nextX, nextY));
                        mField[nextX][nextY] = -1;
                        return true;
                } else if (mField[nextX][nextY] == 1) {
                        return false;
                } else if (mField[nextX][nextY] > 1) {
                        mScore=mScore+10;
                        mField[nextX][nextY] = 0;
                        mSnake.add(new pos(nextX, nextY));
                        mField[nextX][nextY] = -1;
                        addFruite();
                        return true;
                }
```

```
                }

                if((nextX >= mFieldX)||(nextX < 0)) {
                        nextX=0;
                        if (mField[nextX][nextY] == 0) {
                                mField[mSnake.get(0).x][mSnake.get(0).y] = 0;
                                mSnake.remove(0);
                                mSnake.add(new pos(nextX, nextY));
                                mField[nextX][nextY] = -1;
                                return true;
                        } else if ((nextX < mFieldX) && mField[nextX][nextY]
== 1) {
                                return false;
                        } else if ((nextX < mFieldX) && mField[nextX][nextY] >
1) {
                                mScore=mScore+10;
                                mField[nextX][nextY] = 0;
                                mSnake.add(new pos(nextX, nextY));
                                mField[nextX][nextY] = -1;
                                addFruite();
                        return true;
                        }
                }
                else{
                        return false;
                }
        }
        // כיוון דרום
        case DIR_SOUTH: {
                int nextX = mSnake.get(mSnake.size() - 1).x;
```

```
int nextY = mSnake.get(mSnake.size() - 1).y + 1;

if (nextY < mFieldY) {
        if(mField[nextX][nextY] == 0) {
                mField[mSnake.get(0).x][mSnake.get(0).y] = 0;
                mSnake.remove(0);
                mSnake.add(new pos(nextX, nextY));
                mField[nextX][nextY] = -1;
                return true;
        } else if (mField[nextX][nextY] == 1) {
                return false;
        } else if (mField[nextX][nextY] > 1) {
                mScore=mScore+10;
                mField[nextX][nextY] = 0;
                mSnake.add(new pos(nextX, nextY));
                mField[nextX][nextY] = -1;
                addFruite();
                return true;
                }
}
if((nextY >= mFieldY)||(nextY < 0)){
        nextY=0;
        if ( mField[nextX][nextY] == 0) {
                mField[mSnake.get(0).x][mSnake.get(0).y] = 0;
                mSnake.remove(0);
                mSnake.add(new pos(nextX, nextY));
                mField[nextX][nextY] = -1;
                return true;
        } else if (mField[nextX][nextY] == 1) {
```

```
                return false;
            } else if (mField[nextX][nextY] > 1) {
                mScore=mScore+10;
                mField[nextX][nextY] = 0;
                mSnake.add(new pos(nextX, nextY));
                mField[nextX][nextY] = -1;
                addFruite();
                return true;
            }
        }
        else{
            return false;
        }
    }
    // כיוון מערב
    case DIR_WEST: {
        int nextX = mSnake.get(mSnake.size() - 1).x - 1;
        int nextY = mSnake.get(mSnake.size() - 1).y;
        if (nextX >= 0) {
            if (mField[nextX][nextY] == 0) {
                mField[mSnake.get(0).x][mSnake.get(0).y] = 0;
                mSnake.remove(0);
                mSnake.add(new pos(nextX, nextY));
                mField[nextX][nextY] = -1;
            return true;
        } else if (mField[nextX][nextY] == 1) {
            return false;
        } else if (mField[nextX][nextY] > 1) {
            mScore=mScore+10;
```

```
                mField[nextX][nextY] = 0;
                mSnake.add(new pos(nextX, nextY));
                mField[nextX][nextY] = -1;
                addFruite();
                return true;
                }
        }

        if((nextX < 0)||(nextX >= mFieldX)) {
                nextX=mFieldX-1;
                if (mField[nextX][nextY] == 0) {
                        mField[mSnake.get(0).x][mSnake.get(0).y] = 0;
                        mSnake.remove(0);
                        mSnake.add(new pos(nextX, nextY));
                        mField[nextX][nextY] = -1;
                        return true;
                } else if ((nextX >= 0) && mField[nextX][nextY] == 1)
{
                        return false;
                } else if ((nextX >= 0) && mField[nextX][nextY] > 1) {
                        mScore=mScore+10;
                        mField[nextX][nextY] = 0;
                        mSnake.add(new pos(nextX, nextY));
                        mField[nextX][nextY] = -1;
                        addFruite();
                        return true;
                }
                return true;
        }
        else{
```

```
                        return false;
                }
            }
        }
        return false;
    }



    public int getDirection() {
        return mDirection;
    }

    public void clearScore(){
        this.mScore=0;
    }

    public void setDirection(int direction) {
        this.mDirection = direction;
    }

    public int[][] getmField() {
        return mField;
    }

    public int getSnakeLength() {
        return  mSnake.size();
    }

    public ArrayList<pos> getmSnake() {
```

```
        return mSnake;
    }
}
```

לפי דרישותינו יצרנו שני מספרים בעלי POS. נסכם ונבדוק מה יצרנו. ונתחיל מהמחלקה אופי :

- לאחר מכן יצרנו קבועים לטובת הייצוג של הכיוונים ויצירת משתנה בו Y. וציר X ציר **mDirection**. - נשמור את סוג הכוון הנוכחי
- עם קואורדינטות דו ממדיות לכל פלג גוף של נחש - **mSnake** – גוף הנחש (סגמנט).
- מערך דו ממדי שבו כל אלמנט מחזיק קוד של תא בודד - **mField** – שדה המשחק בשטח המסך של משחק. כלומר 1 – מייצג את התא שבו נמצא נחש ברגע נתון. 2 – מייצג את התא ובו נמצא הפרי. 0 – מייצג את תא ובו לא נמצא דבר. אפשרות נוספת אשר הוספנו כדי להקטין כמות הנתונים היא – אפשרות לייצג את קוד 1 לטובת קידוד קיר במשחק.
- הקונסטרוקטור של המחלקה הזאת קובע את 3 דברים היקרים והם – ניקוי שטח המשחק, קביעה של מיקום ראשוני התחלתי לנחש. ובנוסף בעזרת מתודה **addFruite()** אנחנו מוסיפים פרי אחד למשטח המשחק.

במחלקה הזאת קיימת מתודה מסיבית וחשובה:

- **מתודה nextMove()** - כאשר הנחש true - תפקידה להחזיר את אמת **mDirection**. מסוגל לנוע בשטח המשחק לפי הנחיות שנקבע במשתנה בראשית קובעים לאיזה כיוון הנחש חייב לנוע, לאחר מכן לכל כיוון בנפרד נבדקות תכונות הבאות: האם הנחש נתקע בקיר, האם הנחש אכל את הפרי, אם כן אז לקבוע לאיזה כיוון הגוף של הנחש חייב לגדול. כאשר הבדיקות מסתיימות כולן בהצלחה אז מחזירים אמת – **true** ואם לא **false.**

GameActivity

```
package com.snake.lm;

import java.util.List;

import java.util.Timer;

import android.app.Activity;

import android.hardware.Sensor;

import android.hardware.SensorEvent;

import android.hardware.SensorEventListener;

import android.hardware.SensorManager;

import android.os.Bundle;

import android.util.Log;

import android.view.MotionEvent;

import android.view.WindowManager;

import android.widget.Toast;


public class GameActivity extends Activity implements SensorEventListener {


	GameSurface surf;

	Timer t;

	int width, height;

	boolean pause;


	StepUpdater task = new StepUpdater(this);


	SensorManager mSensorManager;
```

```
Sensor mAccelerometerSensor;



float SSX = 0, SSY = 0;
float SX = 0, SY = 0;
boolean firstTime;

// activity מטפלים ביצירת
@Override
public void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	surf = new GameSurface(this);
	this.setContentView(surf);
	t = new Timer();
	height = this.getWindowManager().getDefaultDisplay().getHeight();
	width = this.getWindowManager().getDefaultDisplay().getWidth();

	// accelerometer אתחול
	mSensorManager = (SensorManager)
getSystemService(Activity.SENSOR_SERVICE);
	List<Sensor> sensors =
mSensorManager.getSensorList(Sensor.TYPE_ALL);
	if (sensors.size() > 0) {
		for (Sensor sensor : sensors) {
			if (sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
				if (mAccelerometerSensor == null)
					mAccelerometerSensor = sensor;
			}
		}
	}
```

```
}



//activity הפעלת
@Override
public void onStart() {
        super.onStart();
        pause=true;

        //  הפעלת  טיימר הרענון על מסך
        t.scheduleAtFixedRate(new GraphUpdater(surf), 0, 100);
        // הפעלת טיימר שמעדכן את המיקום של הנחש
        t.scheduleAtFixedRate(task, 0, 500);//changed
        // הרשמה הטופס שלנו כאובייקט של
        // (accelerometer) אשר מקבל את השינויים של החיישן
        mSensorManager.registerListener(this, mAccelerometerSensor,
                        SensorManager.SENSOR_DELAY_GAME);
        this.firstTime = true;
        //activity מעבדים הפסקת

    getWindow().setFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON,
WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
    }


    public boolean onTouchEvent(MotionEvent event) {
        //טיפול במגע
        // pause הפעלת
        StepUpdater new_task = new StepUpdater(this);
      switch (event.getAction()) {
            case MotionEvent.ACTION_UP:
```

```
                if (pause==true){
                        pause=false;
                        Toast.makeText(getApplicationContext(), "Pause",
Toast.LENGTH_LONG).show();
                        new_task.act = task.act;
                        task.cancel();
                        t.cancel();
                        t.purge();
                        return true;
                }
                else {
                        t = new Timer();
                        t.scheduleAtFixedRate(new GraphUpdater(surf), 0, 100);
                        t.scheduleAtFixedRate(new_task, 0, 500);
                        task.act = new_task.act;
                        task.run();
                        pause=true;

                        Toast.makeText(getApplicationContext(), "Resume",
Toast.LENGTH_LONG).show();

                }
                break;
            }
            return true;
        }
```

```
// מעבדים הפסקות activity
@Override
public void onStop() {
	super.onStop();
	// עוצרת  טיימר
	t.cancel();
	t.purge();
	// לא רושמים
	// שינוי החיישן
	mSensorManager.unregisterListener(this);
}

//	@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {

}

// מתודה שקובעת
// (y ו- x) פרמטרים
//accelerometer לפי אינדיקציות
//לאיזה כיוון להזיז את הנחש
private int getDirection(float x, float y) {
	if (Math.abs(x) > Math.abs(y)) {
		if (x > 0) {
			return SnakeGame.DIR_WEST;
		} else {
			return SnakeGame.DIR_EAST;
		}
```

```
            } else {
                if (y > 0) {
                    return SnakeGame.DIR_SOUTH;
                } else {
                    return SnakeGame.DIR_NORTH;
                }
            }
        }



        //טיפול בשינוי כיוון
        //פלאפון בסביבה
//      @Override
        public void onSensorChanged(SensorEvent event) {
            surf.setSomeText("Your score is: "+SnakeLMActivity.GAME_SCORE);

            // קריאות חיישן
            SX = event.values[0];
            SY = event.values[1];

            //אם המשחק כבר מתבצע
            if (!this.firstTime) {
                //   מקבלים מצב פלאפון בסביבה

                float dirX = SX - SSX;
                float dirY = SY - SSY;
                //מגדירים כיוון חדש לנחש
                surf.mField.setDirection(this.getDirection(dirX, dirY));
                //  העברת קואורדינטות פלאפון בסביבה
```

```
                surf.setXY(dirX, dirY);
            } else {
                //אם המשחק רק התחיל
                // עושים כיול של המיקום ההתחלתי של הטלפון
                this.firstTime = false;
                SSX = SX;
                SSY = SY;
            }
        }

        //מתודה זו נקראת מזרם של אחד הטיימרים
        // בשיטה זו ממומשת תנועה של הנחש

        public void Step() {
            // activity אם המהלך נכשל אז לסגור את
            if (!surf.mField.nextMove()) {
                SnakeLMActivity.GAME_MODE=1;
                this.finish();
            }
            //אם הכל בסדר אז מעדכנים את הנקודות
            // activity התחלתית -ב
            else{
                SnakeLMActivity.GAME_SCORE=this.surf.mField.mScore;
            }
        }
    }
```

כעת נפרט ונבין מה קורה כאן

במשתנים נמצאים surf - אוביקט GameSurface, טיימר t, SensorManager, Sensor.

אנחנו משתמשים בהם על מנת להכניס חיים לנחש.

- מתודה **onCreate()** - מאפשרת לבנות שדה משחק (surf) ולשלב אותו עם ה-Activity הראשי. מאפשרת להגדיר טיימר ולאתחל. עוזרת לקבל את נתוני רזולוציית המסך.

- מתודה **getSystemService()** - מאפשרת גישה בעזרת משתנה **mSensorManager** לכל החיישנים המובנים בפלאפון. בעזרת החיישן מגיעים לכיוון הרצוי - accelerometer.

- מתודה **onStart()** - אחראית על תפעול נכון בזמן הפעלת כל Activity. כאן כגוף העוזר להקשיב לכל (**SensorEventListener**) מפעילים צורת הממשק שינוי המתרחש בחיישנים. ובנוסף מכתיבים לפלאפון לא לכבות תאורה במסך בזמן משחק.

- מתודה **onStop()** - אחראית על הפסקת המשחק ועל כיבוי החיישנים המופעלים.

- מתודה **getDirection()** - מקבלים פרמטרים מחיישנים וקובעים לאיזה כיוון בהתייחסות ישירה לחיישנים. כלומר לאיזה y או לציר x צריך לפנות הנחש, לציר צד הפנינו את פלאפון.

- מתודה **onSensorChanged()** - מקבלת נתונים מהחיישנים ובודקת האם המשתמש הפעיל הרגע את המשחק, ואם כן קובעים את צורת הפלאפון בסביבה כברירת מחדל. אחרת מתקנים את הנתונים אשר באים מחיישנים ביחס לברירת המחדל.
  לטובת המשתמש יצרנו אפשרות זו, כדי שיוכל להשתמש בפלאפון בכל כיוון, ולא רק בכיוון מאונך.

- מתודה **step()** - מגדירים פקודה לנחש לנוע לתא הבא, מקבלים תשובה חיובית או שלילית ומחשבים את ביצוע הפעולה הבאה. כלומר תשובה חיובית – המשך תזוזה. תשובה שלילית – נתקענו ונצא למסך הסיום.

SnakeLMActivity

```
package com.snake.lm;

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

```
import android.widget.Button;
import android.widget.TextView;

public class SnakeLMActivity extends Activity implements OnClickListener {

	Button butt;
	TextView tv;

	//activity הפעלת
	//0- הרצה הראשונה
	//1 - תתחיל אחרי שהפסיד
	public static int GAME_MODE=0;

	public static int GAME_SCORE=0;


	public void PauseGame()
	{
		SnakeGame sg = new SnakeGame();
		sg.mScore = 12;
	}
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
	}

	@Override
	public void onStart(){
		super.onStart();

		//טעינת הפריסות השונות
		if (GAME_MODE==0){
		setContentView(R.layout.main);
		butt = (Button) this.findViewById(R.id.button1);
		butt.setOnClickListener(this);
		}
		else
		{
			setContentView(R.layout.lose);
			butt = (Button) this.findViewById(R.id.button2);
			tv = (TextView) this.findViewById(R.id.textView2);
			tv.setText("Your score: "+GAME_SCORE);
			butt.setOnClickListener(this);
		}
	}

	public void onClick(View v) {
		// אם לוחצים על הכפתור, המשחק מתחיל
		Intent i = new Intent(this, com.snake.lm.GameActivity.class);
		GAME_MODE=0;
		GAME_SCORE=0;
		this.startActivity(i);
	}

}
```

ומסנכרנים את Activity כפי שנראה מהקוד אנחנו מעלים את הקובץ בזמן הרצת המתודות הבאות לפי הנתונים הנרשמים במשתנה **GAME_MODE.**

- מתודה **PauseGame()** - שליטה על כפתור הפסקה בזמן משחק
- מתודה **onClick()** - שליטה על רענון מהפסקה וחזרה למשחק
- מתודה **onStart()** - טעינת נקודות למסך סופי שצבר משתמש: "Your score: "

GameSurface

```
package com.snake.lm;

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

public class GameSurface extends SurfaceView {

	SnakeGame mField;

	Bitmap mHead, mTill, mBody, mBg, mFruite;

	String someText = "Enjoy =)";

	float x, y;

	 Bitmap mWall;

	// התקנה של הקואורדינטות החדשות של הפלאפון
	// כדי לצייר עיגולים כראוי על רקע
```

```
public void setXY(float x, float y) {
	this.x = x;
	this.y = y;
}

//  קונסטרקטור שבו אנחנו טוענים
//  ממשאבים bitmap

public GameSurface(Context context) {
	super(context);
	// אנו יוצרים שדה משחקים חדש
	mField = new SnakeGame();
	//טעינה של תמונות
	mHead = BitmapFactory.decodeResource(context.getResources(),
			R.drawable.head);
	mTill = BitmapFactory.decodeResource(context.getResources(),
			R.drawable.till);
	mBody = BitmapFactory.decodeResource(context.getResources(),
			R.drawable.body);
	mBg = BitmapFactory.decodeResource(context.getResources(),
			R.drawable.bg);
	mFruite = BitmapFactory.decodeResource(context.getResources(),
			R.drawable.fruite);
	mWall= BitmapFactory.decodeResource(context.getResources(),
			R.drawable.wall);
}

// מתודה שבה מוגדר הטקסט
public void setSomeText(String someText) {
	this.someText = someText;
}

// מציירים כאן
void drawSnake(Canvas c) {
	int width = c.getWidth();
	int height = c.getHeight();
	int mx = width / SnakeGame.mFieldX;
	int my = height / SnakeGame.mFieldY;
	//bitmap
	Bitmap head = Bitmap.createScaledBitmap(mHead, mx, my, true);
	Bitmap body = Bitmap.createScaledBitmap(mBody, mx, my, true);
	Bitmap till = Bitmap.createScaledBitmap(mTill, mx, my, true);
	Bitmap bg = Bitmap.createScaledBitmap(mBg, mx, my, true);
	Bitmap fruite = Bitmap.createScaledBitmap(mFruite, mx, my, true);
	Bitmap wall = Bitmap.createScaledBitmap(mWall, mx, my, true);

	// מברשת
	Paint paint = new Paint();
	// המעגל גדול בצבע אדום
	paint.setColor(Color.RED);
	//מציירים עיגולים
	c.drawCircle(width / 2, height / 2, width / 4, paint);
	paint.setColor(Color.BLUE);
	c.drawCircle(width / 2 - x * 5, height / 2 + y * 5, width / 10, paint);
	paint.setColor(Color.RED);
	paint.setAlpha(128);
```

```
            // מציירים את שדה משחק עם פרות עליו
            for (int i = 0; i < SnakeGame.mFieldX; i++) {
                    for (int j = 0; j < SnakeGame.mFieldY; j++) {
                            c.drawBitmap(bg, mx * i, my * j, paint);
                            if (mField.getmField()[i][j] ==2) {
                                    c.drawBitmap(fruite, mx * i, my * j, paint);}
                                    if(mField.getmField()[i][j]==1)
                                    c.drawBitmap(wall,mx*i,my*j, paint);

                    }
            }
            paint.setAlpha(0);
            // מציירים את הנחש
            for (int i = 0; i < mField.getSnakeLength(); i++) {
                    c.drawBitmap(body, mField.getmSnake().get(i).x * mx, mField
                                    .getmSnake().get(i).y * my, new Paint());
                    if (i == 0) {
                    c.drawBitmap(till, mField.getmSnake().get(i).x * mx, mField
                                            .getmSnake().get(i).y * my, new Paint());
                    }
                    if (i == mField.getSnakeLength() - 1) {
                            c.drawBitmap(head, mField.getmSnake().get(i).x * mx,
mField
                                            .getmSnake().get(i).y * my, new Paint());
                    }
            }
            // מציירים את הטקסט
            paint.setColor(Color.WHITE);
            paint.setAlpha(255);
            paint.setTextSize(15);
            c.drawText(someText, 50, 50,  paint);
    }
}
```

מתודות פנימיות כדי להיעזר בהן ולהתחיל **SurfaceView** – מחלקה אשר יורשת מ לצייר את שדה המשחק, צורות שמשתמשים בהן במשחק (נחש, קיר, פרי) והגדרת גודל האותיות:

- מתודה **setSomeText()** – הגדרת אותיות במשחק.
- מתודה **drawSnake()** - מציירת צורת הנחש.
- מתודה **Paint()** - מציירת שני עיגולים במרכז האפליקציה לטובת שליטה יותר טובה על אקסלומטר.

כאן ניתן לשלוט בהגדרת צבע גם לשדה, גם לנחש, וגם לכפתורים. ניתן גם להכניס תמונות מוכנות בפורמט *.bmp

GraphUpdater & StepUpdater

```
package com.snake.lm;
import java.util.TimerTask;
public class StepUpdater extends TimerTask {
	GameActivity act;
	StepUpdater(GameActivity s){
		this.act = s;
	}

	@Override
	public void run() {
		act.Step();
	}
{

package com.snake.lm;
import java.util.TimerTask;
import android.graphics.Canvas;
import android.graphics.Color;
public class GraphUpdater extends TimerTask {

	GameSurface surf;
	GraphUpdater(GameSurface surf){
		this.surf = surf;
	}

	@Override
```

```
        public void run() {
                Canvas c = surf.getHolder().lockCanvas();
                if (c!=null){
                        c.drawColor(Color.BLACK);
                        surf.drawSnake(c);
                        surf.getHolder().unlockCanvasAndPost(c);
                }
        }
}
```

שתי מחלקות דומות המכניסות חיים למשחק
הגדרת טיימר לפי רצוננו בעזרת – **TimerTask** -שתי מחלקות מקבלות יורשה מ מתודה **onStart()**.

- **GraphUpdater** - בזמן הרצה מקבל אוביקט **GameSurface** ומצייר את צורות נחש, שדה, פרי, קירות ועוד.
- **StepUpdater** - מקדם לשלב הבא כאשר מחלקה **GraphUpdater** סיימה את השלב הנוכחי. בכך מתאפשר ציור תזוזת הנחש.

ביבליוגרפיה:

Books:

1. Professional Android 2 Application Development (Reto Meier)
2. Sams Teach Yourself Android™ Application Development in 24 Hours **(Lauren Darcey; Shane Conder)**
3. Pro Android 2 (Paperback)

Saits:

http://stackoverflow.com/
http://www.helloandroid.com/
http://habrahabr.ru/
http://startandroid.ru/
http://developer.android.com/index.html
http://www.basic4ppc.com/
http://www.coderanch.com/
http://www.codeproject.com/
http://www.androidhive.info/
http://www.xtensivearts.com/
http://www.abelski.com/
http://developer.android.com/index.html