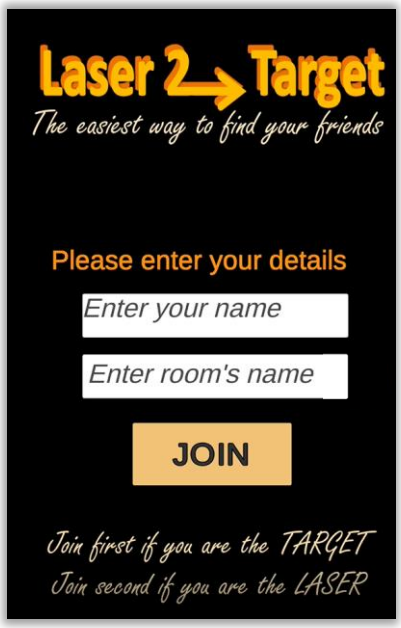
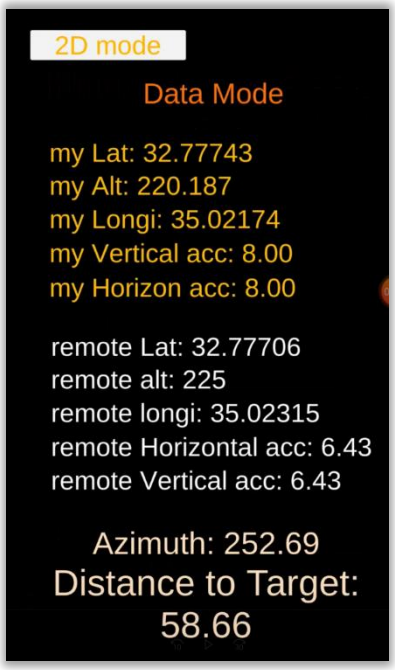
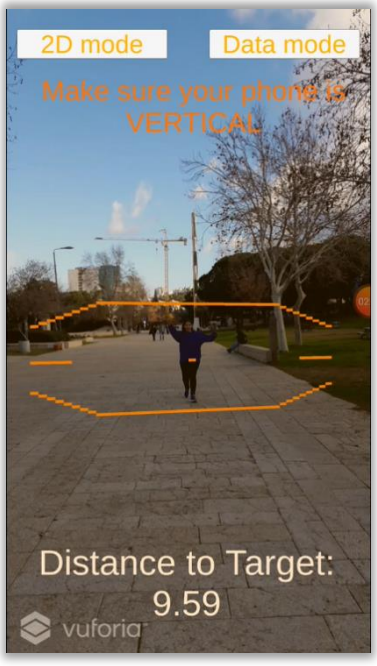
Laser2Target







מבוא

פרויקט זה נולד ביוזמתנו, ולווה ע״י ירון חונן, כפתרון לאנשים שאינם מבינים כלל בניווט ובאומדן מרחקים.

עבור אנשים אלה, אפליקציות ניווט אשר מציגות למשתמש מפה וחץ אינן יעילות שכן התאמת המפה לעולם האמיתי היא משימה קשה עבורם.

הפרויקט שלנו מאפשר לאנשים הנ״ל להשתמש בממשק נוח ואינטואיטיבי אשר מספק להם אך ורק את המידע הנחוץ לצורך הגעה מהירה ליעדם:

1. חץ גדול ובולט עם הכיוון שעליהם להתקדם בו.
2. המרחק למטרה.

כיוון שהמרחק למטרה אינו מועיל רבות לאנשים שאומדן מרחקים אינו בראש מעיניהם- אנו רותמים את טכנולוגית המציאות הרבודה ומשתמשים במצלמת הפלאפון להצגת ״קפסולה״ אשר עוטפת את המטרה בהתאם למרחק הנ״ל, ובכך מאפשרת להבין היכן נמצא היעד.

לפני שהתחלנו בקידום הפרויקט חיפשנו מקורות של קוד פתוח וקוד מסחרי כדי להבין היכן העולם נמצא ביחס למתן פתרונות לבעיה שציינו. מבחינת Open Source לא מצאנו קוד או אפליקציה דומה למה שרצינו ליצור. קיימות אפליקציות המקבלות נ.צ. קבוע והן מציגות ״סיכה״ בכיוון שדרוש להתקדם אליו ומשלבות גם מפה על המסך, אך לא קיימת אפליקציה שמשלבת מציאות רבודה עם מיקום של שני אנשים שמשתנה בזמן אמת.

מבחינת קוד מסחרי- Google מפתחים כרגע פתרון לאפליקציית הניווט שלהם שמשלב מציאות רבודה, אך הפתרון נוגע להגעה ליעדים נייחים וידועים מראש, ואינו כולל ניווט ליעד אנושי שזז בזמן אמת.

יש לציין כי האפליקציה שלנו אינה נותנת מענה לניווט בשטח בנוי מסועף, שכן היא נותנת ״קו ישיר״ למטרה ללא התחשבות במכשולים, אך מנגד היא כן נותנת מענה לסיטואציה שבה ברצונך למצוא יעד אנושי ספציפי כאשר ״המחפש״ ו״היעד״ נמצאים שניהם מחוץ למבנה, ובאותו מרחב פתוח (למשל - המרחב שבין בניינים בטכניון, מתחם פתוח של קונצרט, חוף הים וכו׳).

להערכתנו תחום המציאות הרבודה יתפתח רבות בשנים הקרובות, ובתוך כך אנו מעריכים שבשנים הבאות תופץ אפליקציה דומה במהותה למה שנציג בפרויקט זה.

מוקדש באהבה ל״מחפשים״ באשר הם,

- מיה סאבו ולב טוניק, חורף 2020.

מוטיבציה ומטרה

בפרק זה נפרט על המטרה שלנו, על המניעים לפרויקט, ועל התרחישים השונים שבהם עבודתנו יכולה לתרום למשתמשי האפליקציה.

המטרה שלנו בפרויקט זה היא ליצור פתרון מבוסס מציאות רבודה שמקל על הגעה ליעד אנושי, שנמצא בשטח פתוח עם עוד אנשים, במרחב שבו קשה להסביר היכן נמצא היעד.

לדוגמא: על מנת למצוא מישהו, יכול ה"יעד" להסביר בדרך זו היכן הוא נמצא: "אני נמצא מתחת לעץ \ בצמוד לכיכר" - במרחב עם עצים רבים מסביב, או אנשים נוספים מסביב לכיכר, אך הנחיה זו לא תעזור למצוא את היעד במהירות.

המוטיבציה לפרויקט הגיע מכך ששנינו חווינו לפחות פעם אחת מצב בו מישהו ניסה להסביר לנו כיצד למצוא אותו, וזה לקח לנו הרבה יותר זמן מהצפוי, זאת למרות שהמרחק בינינו לבין היעד היה לכל היותר כמה מאות מטרים.

בנוסף, מניסיוננו האישי, גם כאשר מציינים כי: "אני נמצא X מטרים מימך" הדבר לא תמיד עוזר, לכן היה לנו חשוב לשלב מציאות רבודה בפתרון שלנו, כאמצעי לתרגם את המרחק ליעד לאובייקט גרפי מוחשי שניתן לראות על המסך, ובאמצעותו להבין בדיוק היכן היעד נמצא.

הרצון לפתח את הפתרון הנ"ל התחזק כאשר חברים שיתפו עימנו סיטואציות מביכות כמו המתוארת כאן:

אדם אסף אותם לטרמפ וכתב להם בהודעה "אני מתחת למבנה...". הם חיפשו את האדם הזה מסביב לכל המבנה ולכן התעכבו או לחילופין חשבו שהבינו היכן הוא נמצא אך טעו.

הסיטואציה המוצגת חזרה על עצמה בכמה וריאציות בהן היה רצון להיפגש עם אדם שאינם בהכרח מכירים.

לסיכום, ראוי לציין כי תקשורת בין בני-אדם היא דבר לא פשוט, בעיקר כאשר הדבר נוגע להסבר על מיקום, לכן כפי שפתרון פשוט כמו "BON" (הטופס שהמלצר מדפיס לטבח שבו כתובה הזמנת הלקוח) פתר לחלוטין את בעיית התקשורת במסעדות בין מלצרים לטבחים, לדעתנו הפתרון שלנו יכול לפתור את בעיות התקשורת שנוגעות למפגשים בין אנשים הכוללים הגעה מאחד לשני ומציאת האחר בסביבה.

תיאור המערכת

תיאור הציוד

ציוד קצה עבור משתמשי האפליקציה

בפרויקט זה בדקנו את האפליקציה עבור:

- Samsung S7

זהו אחד המכשירים המהווים "תנאי סף תחתון" לרתימת מציאות רבודה באפליקציות עבור Android.

- דרישת הסף נוגעת אך ורק לפיצ'ר של המציאות הרבודה (שימוש ב- AR mode).

האפליקציה הותקנה ועבדה גם על המכשיר הבא (ללא שימוש במציאות רבודה):

- Samsung S5

כלומר למיטב ידיעתנו, גם מכשירי Android ישנים יחסית שיוצרו החל מ-2015 יכולים להיות "היעד", כיוון שהדבר היחידי שדרוש מהם מבחינת חומרה הוא העברת נתוני מיקום.

ציוד קצה עבור מפתחי האפליקציה

- האפליקציה נבנתה בתוכנה הבאה:
  - Unity 2019.2.10 לינק להורדה: https://unity3d.com/get-unity/update

התוכנה הנ"ל אפשרה לנו לשלב בין כלל רכיבי המערכת, ולבנות אפליקציה לסביבת Android:

1. מודולי קוד\Assets\Packages (העברת נתונים ברשת בין מכשירים, קבלת נתוני חומרת הפלאפון)
2. ממשק משתמש (UI)
3. מנוע גרפי (שליטה על אובייקטים מסוג 2D/3D בסצנה).

- האפליקציה פותחה בסביבת Windows על המחשבים האישיים שלנו.
  - איננו מפרטים על רכיבי החומרה של המחשבים האישיים שלנו שכן הם נקנו לפני מספר שנים ואינם בעלי חומרה שנחשבת טובה כיום (2020), לכן אנו בטוחים כי חומרת המחשב של מפתח עתידי לא תהווה מחסום בדרך לפיתוח האפליקציה.

- להלן פירוט של כלל רכיבי הפלאפון בהם השתמשנו
  - GPS- לטובת קבלת המיקום של המכשיר (המימוש הפנימי שאינו קשור אלינו, משתמש גם במידע על נתוני אנטנות של WIFI לצורך העלאת רמת הדיוק).
  - Gyroscope- תורם להבנת נטיית המכשיר במישור האנכי לקרקע (plane Z), ועל ידי כך מאפשר לנו להתאים את כיוון החץ למטרה גם כשהמכשיר "מסובב".

- Camera- מאפשרת לנו להסתכל על העולם האמיתי דרך המצלמה.

- להלן פירוט של כלל המודולים שהוספנו לפרויקט דרך ה-Asset Store של Unity:

- הקפדנו להשתמש רק במה שהיה חינם.
(קיימות חבילות נוספות שעולות כסף ויכולות לחסוך זמן פיתוח)

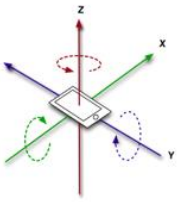
- Photon Multiplayer 2 – אפשר לנו להעביר נתוני מיקום ממכשיר "היעד" למכשיר "המחפש".

https://dashboard.photonengine.com/en-us/account/SignUp

- Vuforia- אפשר לרתום את המציאות הרבודה לטובת הפרויקט.
https://developer.vuforia.com/downloads/sdk?d=windows-30-16-10248&retU

- Log Viewer- אפשר לקבל בצורה פשוטה גישה לנתוני DEBUG על גבי מכשיר הפלאפון בזמן הרצת האפליקציה
https://www.youtube.com/watch?v=UI6qXfYs1Zw&t=139s

- להלן פירוט גרפי של הכלים העיקריים שבהם השתמשנו במהלך פיתוח האפליקציה:

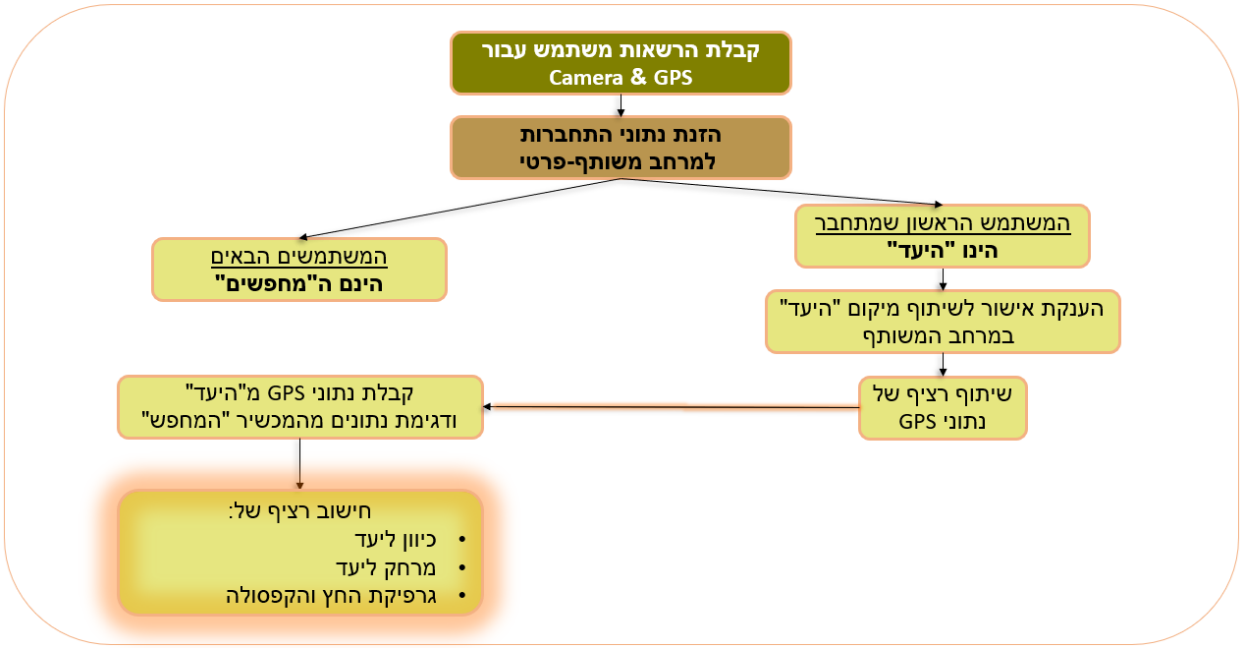

שלבים עיקריים בהקמת סביבת העבודה

1. התקנת Unity והפעלת חשבון משתמש חינם
2. פתיחת פרויקט חדש ב-Unity
3. הוספת החבילות הבאות דרך Asset Store מתוך החלונית של הפרויקט
    a. Photon multiplayer 2
    b. Vuforia
    c. Log Viewer
4. הפעלת חשבון משתמש גם באתרים הבאים:
    a. https://dashboard.photonengine.com/en-us/account/SignUp
    b. https://library.vuforia.com/articles/Training/getting-started-with-vuforia-in-unity.html
5. אישור סביבת עבודה של "מפתח" בפלאפון.

כעת ניתן יהיה לשלב את יכולותיהם של הכלים הנ"ל לצורך שחזור הפרויקט שבנינו.

- חשוב לציין כי במידה ומשתמשים ב-GIT לצורך עבודה משותפת בין מספר מפתחים, ובגרסאות שונות של - Unity\ Visual Studio בין המפתחים השונים, עולה הצורך לפעמים לייבא מחדש את החבילות הנ"ל. זוהי מבחינתנו תקלה של סביבת הפיתוח ולא מצאנו לה פתרון אחר, לכן היה לנו חשוב לציין זאת בעבודה.

Algorithm flow



להלן הסבר מילולי של עיקרי האלגוריתם:

1. במהלך ההפעלה הראשונה של האפליקציה במכשיר- האפליקציה מבקשת הרשאות למצלמה ולנתוני המיקום של המכשיר.
2. <u>סצנה ראשונה-</u> משותפת לכלל משתמשי האפליקציה. בסצנה זו נדרשים המשתמשים להזין שם, ואת שם המרחב הפרטי-המשותף שלהם.
   - המשתמש הראשון שמתחבר למערכת במסך זה יהיה תמיד "היעד", כיוון שהוא זה שמעונין שימצאו אותו.
   - המשתמשים הבאים שיתחברו למערכת יהיו "המחפשים".
3. עדיין בסצנה הראשונה- ה"יעד" ילחץ על כפתור "Find me" ובכך יאשר לכלל המשתמשים במרחב הפרטי-המשותף לעבור לסצנה הבאה.
4. <u>סצנה שניה</u>- בסצנה זו "היעד" שולח את נתוני ה-GPS שלו לכלל ה"מחפשים" באופן רציף.
5. ה"מחפשים" דוגמים נתוני GPS מהמכשיר שלהם, וביחד עם הנתונים שקיבלו מהיעד מבצעים את החישובים הבאים באופן רציף:
   - מה הכיוון ליעד (על סמך נק' GPS ונטיית המכשיר ביחס למישור Z)
   - מה המרחק ליעד (על סמך נק' GPS)
   - כיצד לצייר את גרפיקת הקפסולה והחץ על המסך (בהתבסס על החישובים הנ"ל, ונתוני ה-Gyroscope)
6. בסצנה זו קיימים 3 מסכים אפשריים ביניהם ניתן לעבור בכל שלב. מסכים אלו מופיעים במכשיר של המחפש בלבד. (המסך של מכשיר היעד מכיל הודעה כי המחפש נמצא בדרך אליו).

- מסך 2D mode: המסך מכיל את החץ המכוון למטרה, החץ מראה את הכיוון בו יש ללכת על מנת להגיע ליעד. בנוסף, מצוין בו המרחק ליעד וכן כיצד יש להחזיק את הטלפון על מנת שחישוב המרחק והכיוון יעבוד כראוי (החזקת הטלפון במקביל לקרקע). כאשר נמצאים במרחק של פחות מ-60 מטרים ליעד, עולה הודעה למסך בה מצוין כי רצוי לעבור למסך AR mode על מנת לראות את הקפסולה מסביב ליעד.
- מסך AR mode: במסך זה רואים דרך מצלמת הפלאפון את הסביבה בה נמצאים והקפסולה מופיעה מסביב ליעד אותו מחפשים. בנוסף, מצוין בו המרחק ליעד וכן כיצד יש להחזיק את הטלפון על מנת שחישוב המרחק והכיוון יעבוד כראוי (החזקת הטלפון במאונך לקרקע).
- מסך Data Mode: בו ניתן לראות את פרטי ה-GPS המלאים של 2 המכשירים, טווח דיוק של הנתונים, זווית Azimuth והמרחק ליעד.

- להעמקה בשלבים השונים ובאתגריהם יש לפנות לפרק הבא- "אתגרים עיקריים".

אתגרים עיקריים

כללי

האתגרים העיקריים שהיו לנו במהלך כתיבת האפליקציה כוללים את היכולת לקבל תוצאה טובה ופשוטה למשתמש בכל שלב באלגוריתם, ולהצליח לשלב בין כל החלקים השונים של המערכת, תוך איזון הזמן שנדרש לקיום הבדיקות ההכרחיות במרחב הפתוח והזמן שנדרש לצורך קידום המימוש.

נדרשנו ללמוד קטעי דוקומנטציית קוד שנמצאים בתוך תיאור של אובייקטים ופונקציות שונות כיוון שלא היו מדריכים או סרטונים בנושא היכולות השונות שרצינו לממש.

בנוסף לכך נדרשנו להתעמק בנושאים הבאים: שיתוף נתוני זמן אמת בין פלאפונים, ניווט, יכולות ושגיאות GPS, רוטציות של דו ותלת מימד (כלומר שילוב מסכי אפליקציה דו מימדית עם מסכים שמשלבים מצלמה וAR).

פתרון לאתגר מסוים נראה היה פשוט אך בדיעבד, כדי להגיע אליו נדרשנו להעמקה תאורטית כדי להבין כיצד להגיע לתוצאה הרצויה, ואז למחקר בקוד כדי להבין כיצד לממש את התאוריה, ואז לניסוי וטעיה בקוד, ורק לבסוף הצלחנו להגיע לתוצאות שמבחינתנו היו מדויקות מספיק כדי להתקדם לאתגר הבא.

אתגר נוסף עיקרי שראוי לציון הוא ש"בדיקת האפליקציה" חייבה אותנו לצאת מחוץ למבנה כדי לבדוק את התרחישים השונים ואת רמות הדיוק. כמות הזמן שהושקעה בבדיקות בחוץ הייתה רבה והכרחית בעינינו, דבר חיובי מצד אחד שכן כך וידאנו שהמערכת באמת עובדת, ומנגד אתגרה אותנו שכן האטה את הקצב שבו מימשנו את הפרויקט.

שיטת שיתוף נתוני המיקום

בבואנו להחליט כיצד נעזור לאנשים למצוא אחד את השני, שקלנו האם עדיף לנו להשתמש באותות ה-WIFI וה-BT שמכשירים מסוגלים לשדר.

בחרנו שלא להשתמש ב-Bluetooth\Wifi אלא להעדיף תקשורת שמבוססת על העברת נתונים בין פלאפונים ברשת האינטרנט, כיוון שהשיטות האחרות מגבילות אותנו במרחק שניתן לשתף נתונים בין שני מכשירים.

המטרה שלנו הייתה לעזור לאנשים למצוא אחד את השני במרחקים של כ-300 מטר ומטה, ולכן במרחקים כאלה לא ניתן להסתמך על WIFI/BT.

בחירת כלי הפיתוח

1. פלטפורמה כללית- בבואנו להחליט באילו כלי פיתוח נשתמש, ביצענו מחקר מקדים, אשר בסופו הבנו ש-Unity תהיה פלטפורמה טובה וחינמית שבה ניתן יהיה לשלב גם יכולות של בניית ממשק משתמש, גם מודולים שונים של תקשורת אינטרנט ויכולות מציאות רבודה, וגם לשלב קוד משל עצמנו.
2. שיתוף נתוני זמן אמת ברשת- בחרנו בכלי שנקרא Photon בתור המערכת שאחראית על יצירת "החדרים" (המרחבים הפרטיים-משותפים). הדרך לבחירה בכלי זה לא הייתה פשוטה, שכן בהתחלה ניסינו להשתמש ב-FireBase, אך לבסוף נטשנו כלי זה כיוון שהבנו שהוא אולי מקיף יותר עם יותר יכולות, אך הוא אינו מיועד למטרה של "משחקים מרובי משתתפים", בעוד ש-Photon מיועד בדיוק למטרה שחיפשנו- שיתוף מהיר של

נתוני זמן אמת בין משתתפים של אותו "משחק". בנוסף ל-Photon הייתה חבילה שניתן היה לייבא לתוך Unity כך שהממשק בין שני הכלים היה טבעי יותר.

3. יכולות מציאות רבודה- בחרנו להשתמש ב-Vuforia כיוון שיש לה ממשק טבעי שמתחבר ל-Unity ע"י חבילה שניתן לייבא לסצנה. בהתחלה רצינו להשתמש ב-ArCore של Google אבל החיבור של Unity ל-Vuforia הרגיש יותר טבעי והייתה אליו דוקומנטציה ידידותית יותר.

אתגרי שיתוף המיקום בין המכשירים

1. איזון עם חיי סוללה- במהלך העבודה עם Photon היינו צריכים להחליט על איזון מסוים בין הרצון לעדכן נתונים בזמן אמת לרצון למזער עד כמה שניתן את צריכת הסוללה של המכשירים, שכן הבנו שככל שנשתף יותר נתונים בין המכשירים, כך הדבר ישפיע יותר על הסוללה שלהם.

   בעקבות כך הגענו להחלטה שרק מכשיר "היעד" ישלח את נתוניו למכשיר "המחפש". כאשר הדבר היחידי החשוב יהיה שמכשיר "היעד" יהיה זה שיאשר סופית לחשוף את מיקומו בפני שאר ה"מחפשים".

2. שיתוף רציף ומדויק- אתגר נוסף שלקח לנו זמן לעלות על פתרונו הוא שכאשר שלחנו את נתוני המיקום בתור Float, הם לא עודכנו במהירות שציפינו, כלומר אם אחד המכשירים זז, וציפינו שכלל המערכת תעדכן את חישוביה, הדבר לא קרה. ברגע ששינינו את הטיפוס שנשלח בין המכשירים ל-String הכל הסתדר. לא מצאנו לכך סימוכין ברשת, אך להערכתנו כיוון שהמספרים ששלחנו השתנו לפעמים בסיפרה ה-6 אחרי הנקודה העשרונית, המערכת כנראה לא חשבה שזה שינוי משמעותי, אך בשבילנו זהו שינוי מאד משמעותי כי הסיפרה ה-6 מייצגת שינוי במטרים.

3. Encapsulation of transferred GPS data through Photon- אנו ניגשים למידע על נתוני המיקום דרך אובייקט שנקרא lastData. אובייקט זה מבצע אינקפסולציה לכל נתוני המיקום הדרושים לנו, ואף למספר נתונים שאינם דרושים לנו. בדיעבד אולי היה "נקי" יותר להעביר את כלל האובייקט ברשת, אך כיוון שבנינו את המערכת בצורה הדרגתית ולא תכננו להשתמש בכל המידע שבתוך האובייקט, החלטנו להעביר ברשת רק דברים ספציפיים כגון- נתוני רוחב, אורך, גובה, ורמות דיוקים לחלק מהנתונים הנ"ל.

חשיבות הצגת נתוני ה-GPS למשתמש

התחבטנו עם השאלה האם דרוש להציג למשתמש את נתוני ה-GPS עצמם. בסופו של דבר החלטנו שדרוש להציג אותם במסך נפרד, כדי לאפשר את הגישה למשתמשים שירצו לראות דווקא את הנתונים המספריים של נתוני המיקום ונתוני השגיאות. הדבר יכול להועיל למשל לאנשים שמתעסקים בניווט קבוצתי ורוצים לוודא בזמן אמת שהמידע הגרפי המוצג להם תואם למידע המספרי.

חשוב לציין כי ספציפית מניסיוננו השגיאה האופקית והאנכית זהות, אך קיימים מיקומים בעולם שבהם השגיאות שונות. בנוסף חשוב לנו לציין כי כיוון שהאפליקציה שלנו מתבססת על נתוני מיקום שמתקבלים מהמכשיר עצמו, קיימים גורמים שאינם בשליטתנו אשר משפיעים על הדיוק של האפליקציה שלנו, למשל:

- עננות
- פריסת הלווויינים ומיקומם ביחס למכשיר

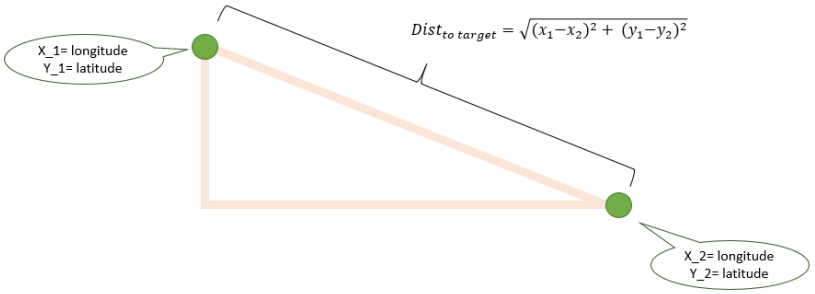
להלן לינקים למידע נוסף:

- https://weather.gladstonefamily.net/gps_elevation.html
- לינק עם סרטון: https://www.youtube.com/watch?v=h1jXqTRto1Q

חישוב המרחק למטרה

חישוב המרחק היה נושא מאד חשוב אצלנו, כיוון שיש שתי דרכים לחשבו, והיינו צריכים להכריע באיזו דרך לממש את חישוב המרחק.

1. הדרך הכי פשוטה היא להסתכל על הבעיה שלנו כבעיה עם שני צירים, כאשר נתונות לנו שתי נקודות, וע"י משפט פיתגורס נחשב את "אורך היתר" אשר שקול למרחק מהמכשיר המחפש ליעדו.



2. הדרך השנייה התייחסה לחישוב המרחק כשתי נקודות שנמצאות על ספירה (Sphere) שמוטלת על מישור.

דרך זו משתמשת בנוסחה הבאה לחישוב המרחק: (מקור: ויקפדיה- מרחק גיאוגרפי)

$$D = R\sqrt{(\Delta\phi)^2 + (\cos(\phi_m)\Delta\lambda)^2},$$

where:

$\Delta\phi$ and $\Delta\lambda$ are in radians;

ϕ_m must be in units compatible with the method used for determining $\cos(\phi_m)$.

To convert latitude or longitude to radians use

$1^\circ = (\pi/180)$ radians.

הערות בנוגע לנוסחה:

- הנחה: המשטח של כדור הארץ בין שתי הנקודות הינו שטוח.
- קיימת התייחסות עבור המרחק המשתנה שבין קווי האורך. ניתן לשים לב בתמונה כי ככל שמתקרבים יותר לחלקו העליון או התחתון של כדור הארץ המרחק שבין הקווים האנכיים קטן.
- לא הצלחנו למצוא הוכחה מתמטית עבור נכונות הנוסחה, אך כן מצאנו שזוהי וריאציה על נוסחת פיתגורס, וכי השימוש בנוסחה זו נפוץ בחישובי מרחק שנעשים במחשב.

3. איזה מהנוסחאות מדויקת יותר?

כדי להבין באיזה מהנוסחאות עדיף לנו להשתמש, ביצענו ניסויים במחשב ובשטח כדי לקבל תוצאות אמינות עד כמה שניתן.

הניסויים במחשב כללו הצלבת נתונים עם אתרים שמחשבים מרחק בין שתי נקודות גיאוגרפיות שנמצאות במרחק שקטן מ-500 מטר אחת מהשנייה.

הניסויים בשטח התבצעו באופן הבא: מדדנו באופן ידני 65 מטר, ואז הרצנו את האפליקציה במצב "DEBBUG" עם הדפסות המרחק של שני סוגי החישובים.

לפי התוצאות, במרחק של 65 מטר שני סוגי החישובים הראו תוצאות זהות, בטווח הבדל של עד 2 מטר.

כלומר לטווח זה שני החישובים נתנו תוצאות זהות, אך בחרנו להמשיך ולהשתמש בנוסחת הספירה (סעיף 2).

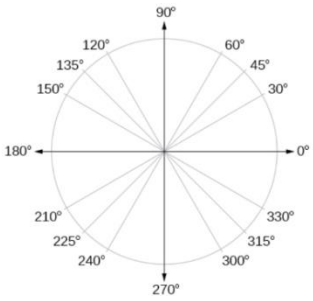
בפועל אנו מבצעים את שני סוגי החישובים כדי שבמידת הצורך נוכל לשנות בין שתי שיטות החישוב בלחיצת כפתור, במידה ונבין בהמשך שצורת חישוב אחרת עדיפה על רעתה, או שנחליט לקחת "ממוצע" על סמך שתי שיטות החישוב.

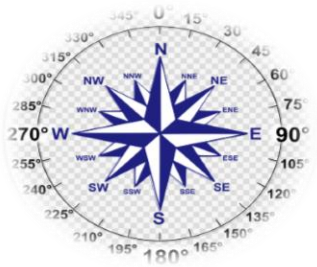
חישוב זווית ההתקדמות למטרה

חישוב הזווית למטרה גם כן היווה אתגר בפני עצמו, שכן קיימת אי התאמה בין מערכת הצירים בעולם האמיתי למערכת הצירים ב-Unity.

לדוגמא: בעולם האמיתי (במדינת ישראל) הים נמצא במערב, כלומר בזווית של 270 מעלות. כאשר במידה ונחזיק מחשב לכיוון צפון, הוא יאמר שהים נמצא בזווית של 180 מעלות מאיתנו. השוני נובע מכך שבעולם הניווט זווית ה"אפס" היא לכיוון צפון, והמעלות עולות עם כיוון השעון, אך במחשב זווית ה"אפס" מתחילה במקביל לציר ה-X, ועולה **נגד** כיוון השעון.

להלן הזויות בעולם האמיתי (מימין) , וב- Unity (משמאל)





כלומר נדרש:

1. לחשב זווית Bearing שזוהי זווית ממקום מסוים למטרה, לפי זווית בעולם האמיתי.
2. לעשות מניפולציה מתמטית כדי להפוך את התוצאה שקיבלנו, לזווית שתטה את החץ שלנו לכיוון המתאים.

חישוב ה-Bearing ייעשה באופן הבא: (מקור: GISMAP)

Bearing of two points

Formula to Find Bearing or Heading angle between two points: Latitude Longitude is

β = atan2(X,Y),

For variable **Y** = **sin(toRadians(lo2-lo1)) * cos(toRadians(la2))**

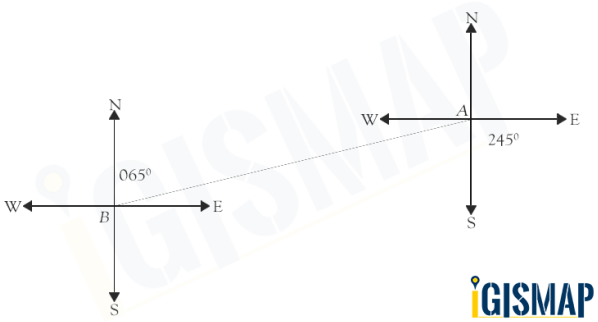
and variable **X** = **cos(toRadians(la1))*sin(toRadians(la2)) – sin(toRadians(la1))*cos(toRadians(la2))*cos(toRadians(lo2-lo1))**

כאשר (יש לשים לב כי ה-X וה-Y התחלפו במיקום בנוסחה הבאה, מקור: ויקפדיה):

$$\operatorname{atan2}(y,x) = \theta = 2\frac{\theta}{2} = 2\arctan\frac{y}{\sqrt{x^2+y^2}+x}\,.$$

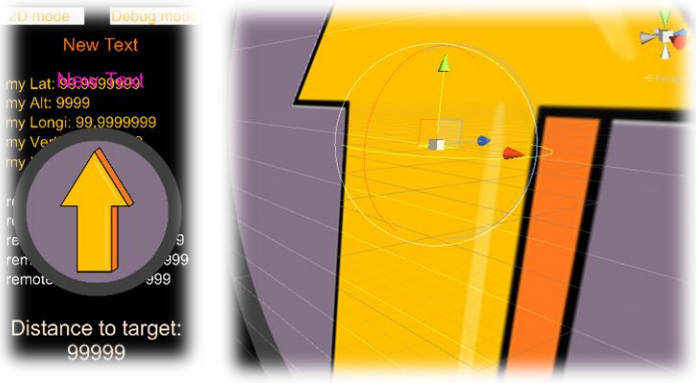
כיוון שנושא זה מסובך לטעמנו, נסביר את התהליך המבוצע ע"י דוגמא שמפרקת לשלבים את התהליך שדרוש לעשות החל מקבלת נתוני המיקום ועד להטיית החץ למטרה.

לדוגמא: נניח כי לפי הנוסחה הנ"ל קיבלנו כי **בעולם האמיתי,** זווית ה-Bearing מאיתנו (מהנקודה השמאלית) ל"יעד" (לנקודה הימנית) הינה 65 מעלות.



כעת, לאחר שקיבלנו את זווית ה-Bearing דרוש לגרום לחץ שלנו להסתובב לכיוון הרצוי.

החץ נמצא במצב ברירת המחדל שלו כלפי "מעלה" במסך, כאשר הסיבוב שנפעיל יתקיים במישור ה-Z (המישור שמסומן בכחול)



בהנחה וראש המכשיר שלנו נמצא "בדיוק לכיוון צפון", אז העדכון היחידי שדרוש לעשות הוא החישוב

```
brng = 360 - brng;
```

החישוב הנ"ל מתבצע כדי לפצות על הפער בין שתי מערכות הצירים השונות שפירטנו עליהן למעלה, והוא יביא את החץ לזווית הרצויה:

- השלב הבא מתבצע תמיד, כיוון שלא ניתן להניח שראש המכשיר יפנה תמיד לצפון הגיאוגרפי, והוא אחראי על התאמת כיוון החץ לכיוון המטרה, תוך התחשבות בסטייה של ראש המכשיר מהצפון הגיאוגרפי:

```
bearing_z+ Input.compass.trueHeading
```

כעת לאחר שהשלמנו את הדוגמא, נציין עוד מספר נקודות בנוגע לאתגר חישוב זווית ההתקדמות למטרה:

- הצפון האמיתי משורת הקוד האחרונה מחושב באופן פנימי על ידי שימוש בחומרת המכשיר (לדוגמא: הג'יירוסקופ), לכן ככל שהמכשיר חדש יותר, הזווית שנקבל לכיוון המטרה תהיה מדויקת יותר, כיוון שהמכשיר ידע לעדכן על סטיה מדויקת יותר מהצפון הגיאוגרפי.
- בפועל, חישוב הזויות ב-Unity מתבצע ע"י אובייקט ששמו QUATERNION. כיוון שזהו נושא מאד רחב, עמוק ומסובך, שעל פי כל ההמלצות שקראנו באינטרנט עדיף לא להתעסק בו ישירות אלא אך ורק דרך הפונקציות שעוטפות אותו, נסביר באופן שטחי בלבד כיצד השתמשנו בו (לינק שמסביר לעומק על אובייקט זה):
  - להלן שורת הקוד שהשתמשנו בה בנושא זה, אשר מטרתה להביא את אובייקט המצפן לזווית הרצויה:

```
compass.transform.rotation = Quaternion.Slerp(compass.transform.rotation, Quaternion.Euler(0f,0f, bearing_z+ Input.compass.trueHeading), 1f);
```

באופן כללי, פונקציית ה-Slerp, מאפשרת לסובב את החץ באופן שייראה רציף ונעים לעין, כאשר היא לוקחת את הרוטציה הנוכחית, ואת הרוטציה הרצויה, ומחזירה "אינטרפולציה ליניארית" בין שני הערכים הללו בהתאם לפרמטר השלישי. בתמונת מסך זו כאשר הפרמטר השלישי הינו 1, מוחזרת רוטציית היעד.

- בחרנו להשתמש בצפון הגיאוגרפי, ולא בצפון המגנטי, כיוון שכשמתעסקים עם מפות רוצים את הצפון של המפה.
- הסיבה שבחרנו להשתמש באובייקט "המצפן" בפלאפון אשר מחשב את הסטייה של ראש המכשיר מהצפון הגיאוגרפי, ולא להשתמש בג'יירוסקופ באופן ישיר, נובעת משני שיקולים:

1. הראשון והעיקרי הוא שאם המכשיר נותן כלי טבעי הנותן מענה לבעיית זיהוי זווית ראש המכשיר ביחס לצפון- זאת סיבה מעולה להשתמש בו.
2. השיקול השני בחשיבותו, אך עדיין ראוי לציינו, הוא שהג'יירוסקופ במכשיר שבדקנו לא נתן קריאות מדויקות כלל כאשר ניגשנו לנתוניו באופן ישיר. דבר זה עלול לנבוע מהמכשיר בו השתמשנו (היה לנו רק מכשיר אחד עם חומרה מספיק עדכנית לבניית אפליקציה של מציאות רבודה). "השפעת צד" נוספת שנבעה מכך היא שבאפליקציה שלנו חייבים להחזיק את הפלאפון מקביל לקרקע כדי שהמצפן יהיה מדויק. בהשוואה לאפליקציית הניווט של Google בה ניתן להחזיק את הפלאפון גם בצורה אנכית לקרקע. כיוון אפשרי שהיה מאפשר לנו לשמור על המצפן מדויק גם כשהפלאפון אנכי לקרקע הינו שימוש ב- Ground plane (בפרק "אתגר הצגת הקפסולה" יוסבר עליו ועל הסיבה שלא השתמשנו בפיצ'ר זה).

אתגר הצגת הקפסולה

הקפסולה מבחינתנו היא האובייקט של "המציאות הרבודה", והיא נועדה לאפשר לאנשים שלא מבינים באומדן מרחק להבין היכן נמצא בערך היעד שלהם, בכך שהיא "עוטפת את המרחב שתואם למיקום המטרה".

בבואנו לבנות את הקפסולה, נתקלנו בבעיות הבאה:

1. כיצד דואגים שהקפסולה תעטוף אדם?
2. כיצד דואגים להתחשב בשגיאות המיקום של שני המכשירים?
3. כיצד דואגים שהקפסולה תתעדכן בהתאם למרחק המשתנה באופן רציף?
4. כיצד לוודא שהקפסולה תהיה בצמוד ומעל למישור הקרקע?
5. האם "לנעול" את הקפסולה על המיקום המשוער של היעד, או שאולי עדיף שהקפסולה תמיד תהיה במרכז המסך בהתאם לכיוון שאליו מסתכל המשתמש?

את רוב הבעיות הנ"ל פתרנו ע"י ניסוי וטעיה שנגעו לכיוון פרמטרי ה-Local Scale ו-Local Transform.

ראשית כל, החלטנו שגובה הקפסולה לא חייב להיות מותאם לגובה של אדם ספציפי. הוא יכול "לעטוף" אדם שגובהו כמטר, והוא יכול לעטוף רק את רגליו של אדם שגובהו כ-2 מטר. הבנו שבעיה זו פחות עיקרית, ולכן לא התעקשנו על הגובה המדויק.

שנית, הבנו שחשוב מאד שהשגיאה המצטברת של נתוני המיקום משני המכשירים תבוא לידי ביטוי בקפסולה, שכן במידה והשגיאות גדולות, הדבר משפיע באופן ישיר על המרחב שבו יכול להימצא "היעד". לכן החלטנו שרוחב הקפסולה יתעדכן באופן רציף בהתאם לשגיאה המצטברת של שני המכשירים, אך קבענו Clamp (פונקציה תוחמת) כך שרוחבה לא יעלה על 20 מטר. במצב שבו השגיאה גדולה מדי- האפליקציה תצטרך להציג למשתמש כי נתוני המיקום אינם מאפשרים תוצאה אמינה. (הסבר נוסף בפרק "כיוונים עתידיים")

הבעיה השלישית, התבררה כבעיה הכי עיקרית שלנו.

Vuforia לא מספקת מנגנון אוטומטי שניתן לספק לו פעם אחת "מרחק" והוא לבד יעדכן את המיקום של האובייקט בהתאם להתקדמותנו או התרחקותנו. במינוח מקצועי-ערך הקפסולה בציר ה-Z לא התעדכן כאשר התקדמנו או התרחקנו מהיעד.

לצורך פתרון נושא זה, החלטנו לעדכן באופן רציף את ערך ה-Z, עם שינוי אחד עיקרי ופשוט בדיעבד, שלקח הרבה זמן להגיע אליו מבחינת הבנה:

לשים את הקפסולה כ-Child של ה-AR camera, ולעדכן את ערך ה-Z של הקפסולה ב-**Local Transform.**

השינוי פתר לנו את בעיית העדכון הרציף, אך כיוון שלא היה לנו ניסיון ב-Unity לפני כן, לא ידענו שיש מערכת צירים נוספת (מערכת מקומית), לכן ההגעה לפתרון זה לקחה זמן רב.

ברגע שבעיה זו נפתרה, שמנו לב לבעיה נוספת שכעת מאד בלטה:

תיקנו את ערך ה-Z, אך ערך ה-Y לא תאם למציאות, כלומר הקפסולה נראתה לפעמים מתחת לאדמה, או גבוה בשמיים, ולא בגובה הרצוי מעל מישור הקרקע.

לצורך פתרון אתגר זה- ביצענו את הניסוי הבא:

מדדנו באופן ידני 65 מטר, כי הנחנו שזה המרחק הראשון הסביר שבו אנשים יפעילו את "המציאות הרבודה" כדי להבין טוב יותר היכן "היעד" נמצא.

בנינו אפליקציית עזר, ובעזרתה בדקנו איזה ערך של Y הכי מתאים למטרתנו, להלן המחשה:

אחרי שמצאנו את ערך ה-Y הכי מתאים, הבנו שהיחס בין המרחק ליעד לבין הגובה הרצוי של ציר ה-Y המקומי של הקפסולה הינו "מינוס 13":

$$\frac{dist_{to\ target}}{Y_{best\ aprioric\ value}} = \frac{65}{-5} = -13$$

לכן בבואנו להחליט איזה ערך לשים לציר ה-Y המקומי עבור מרחקים אחרים מהיעד, אנו מבצעים את החישוב הבא:

$$Y_{value} = \frac{dist_{to\ target}}{-13}$$

זהו פתרון פשוט אשר שיפר באופן ניכר את איכות הקפסולה שלנו.

ראוי לציין כי לעת כתיבת שורות אלה, בהם אנו מסכמים את הפרויקט, גילינו כי Vuforia הוציאה פיצ'ר שפותר בדיוק את בעיית "המיקום בציר ה-Y".

פיצ'ר זה מחפש אוטומטית מישורים אנכיים (כגון המישור שאנחנו הולכים עליו), ואז מאפשר להניח עליו אובייקטים בגובה שהמשתמש מבקש ממנו. כפי שצוין מצאנו פיצ'ר זה בסוף העבודה על הפרויקט ולכן רק בדקנו באופן ראשוני את השימוש בו, אשר נראה מבטיח.

הבעיה האחרונה שהצגנו, בנוגע לדילמה על "נעילת" מיקום הקפסולה במיקום המשוער של היעד, גם כן הייתה החלטה לא פשוטה שנדרשנו לקבל. לבסוף החלטנו שלא כדאי לנעול את מיקום הקפסולה, וכי עדיף שהיא פשוט תהיה תמיד במרכז מסך המצלמה, מהסיבה שאיננו יכולים להבטיח שהאפליקציה תעבוד בצורה טובה בהתבסס על נתוני המיקום (הוסבר במעלה העבודה על בעיות בנתוני המיקום שאינם בשליטתנו). היתרונות בבחירה שעשינו, הם שכעת הממשק הרבה יותר אינטואיטיבי למשתמש- כלומר ברור יותר כי הקפסולה עוטפת את המרחב שנמצא באזור היעד (בהתאם למרחק שהיעד נמצא מאיתנו), אך בפועל המשתמש יכול להסתכל עם הקפסולה לכל כיוון שיחפוץ, ולהבין מה סביבו נמצא במרחק הרלוונטי. באופן זה, אם מישהו נמצא על "גבעה" קטנה או על "עמק" קטן, המשתמש עדיין יוכל להגביה או להנמיך מעט את המצלמה ולהתביית על היעד. (במידה והיינו נועלים את מיקום הקפסולה, לא היינו מאפשרים למשתמש את החופש הזה, והתוצאה במצבים אלה הייתה מונעת את עטיפת היעד- נתוני הגובה של המכשירים לא היו מדויקים מספיק).

הדגשת אתגרי ממשק המשתמש (UI)

אחד הדברים הכי חשובים באפליקציה שלנו הוא ממשק המשתמש. הקדשנו הרבה מחשבה לדרך שבה כל מסך יראה.

במסך ההתחברות לאפליקציה יש להזין את שם המשתמש ואת המרחב בו הוא נמצא. כדי שהחיפוש יוכל להתחיל, על כל המשתמשים להתחבר לאותו המרחב (כלומר להזין שם מרחב זהה).

בגלל שלפני ההתחברות, כל אחד מהמשתתפים יכול להיות ה"מחפש" או ה"יעד", במסך זה יש הבהרה לגבי סדר ההתחברות של המשתמשים (המשתמש הראשון שיתחבר יהיה היעד והשאר יהיו המחפשים). לשם הדגשה, לאחר ההתחברות, מופיע לכל אחד מהמשתתפים מהו תפקידו LASER כלומר המחפש או TARGET כלומר היעד.

בנוסף, לאחר שהמשתמש הראשון התחבר, יהיה עליו להקיש על לחצן חדש שהתווסף למסך (רק אחרי ההתחברות של המשתמש שהוא היעד) על מנת לאשר את התחלת העברת נתוני המיקום שלו לשאר המכשירים. במידה ואין עוד משתמשים מחוברים לאותו המרחב, קופצת הודעת שגיאה המתריעה כי המחפשים עדיין לא הצטרפו.

במסך השני של האפליקציה, מתבצעת האינטראקציה לגבי תהליך החיפוש של היעד.

במסך של היעד, מופיעה הודעה כי מתבצע תהליך חיפוש שלו על ידי המשתמש השני וכי יש להמתין. (בעתיד, ניתן לחשוב על תוספות למסך זה – למשל שיראו גם אצל היעד מה המרחק שלו מהמחפש, שהוא יוכל לקבל התראה כאשר המחפש קרוב וכו').

במסך של המחפש רצינו שהוא יקבל כמה שיותר מידע על מנת שהחיפוש יהיה יעיל ואפקטיבי. מצד שני, לא רצינו שהמסך יהיה עמוס ולא מובן. באופן תמידי מופיע בכל המצבים המרחק של המחפש מהיעד (שלמעט החץ המכוון, הוא הדבר השני בחשיבותו מבחינתנו). בנוסף, באופן ברור, מיקמנו בחלקו העליון של המסך את הכפתורים בעזרתם ניתן לעבור בין המצבים השונים.

במצב 2D, הדבר שרצינו שיתפוס את מירב המקום הוא החץ המכוון (גודלו כמעט מחצית מהמסך) כדי שיתפוס את העין ויהיה האובייקט בו מתמקדים. בנוסף בחרנו ב-2 פרטי מידע חשובים נוספים שיופיעו. אופן החזקת הפלאפון על מנת להבטיח עדכון רציף ומדויק של נתוני המיקום והודעה שקופצת למסך במידה והמרחק ליעד אינו רב (במקרה שלנו 60 מטרים) כדי לבקש מהמשתמש לעבור למצב AR.

במצב AR, המסך כולו הוא "שידור" של מצלמת הפלאפון כדי לאפשר לראות את הסביבה כולה. תחילה הוספנו צבע ("פילטר") למסך אך החלטנו לבטל אותו ולאפשר תצוגה חלקה ונקייה ככל הניתן של המרחב. גם כאן הוספנו המלצה על אופן החזקת הפלאפון. את כל הנתונים במסך זה השתדלנו למקם בתחתית המסך או בחלקו העליון כדי להשאיר כמה שיותר שטח פנוי למצלמה ולקפסולה. תחילה, כאשר המחפש נמצא פחות מ-10 מטרים מהיעד, קפצה הודעה על כך במסך. הבנו כי הודעה זו היא מיותרת מכיוון שבמצב זה המחפש כבר רואה את היעד שלו בבירור, וההודעה מסתירה חלק מסך חשוב ללא צורך ממשי.

את צורת הקפסולה הסופית בחרנו לאחר בדיקה של סוגי AR שונים כדי לוודא שיהיה ברור שזוהי המטרה אליה אנחנו מכוונים להגיע, וגם כדי שנוכל לראות דרכה את האדם אותו אנחנו מחפשים. מצד שני כן רצינו שהיא תבלוט במסך ולכן בחרנו את צבעה בהתאם.

כמו שנכתב לעיל, רצינו לאפשר למשתמש לקבל פרטים נוספים על הנתונים אותם הפלאפון אוסף אך לא רצינו להעמיס על המסכים הקיימים פרטים לא הכרחיים. לכן יצרנו מצב נוסף, מצב Data בו יש פרטים מלאים של המחפש ושל היעד (קו גובה, קו אורך, קו רוחב, טווח שגיאה של הנתונים וזווית על גבי מפה בין 2 המשתמשים).

מסקנות- עיקרי יכולות ומגבלות האפליקציה

יכולות- האפליקציה שלנו מסוגלת לעזור לאנשים למצוא אחד את השני בצורה נוחה, אשר מבטלת את הצורך בשיחות המנסות לתאר (באופן לא אפקטיבי) את המיקום של כל אחד מהצדדים המשתמשים בה. היא עובדת בצורה אופטימלית כאשר המשתמשים נמצאים באותו גובה, וכאשר ל"מחפש" יש פלאפון שתומך במציאות רבודה.

מגבלות- האפליקציה שלנו תלויה בנתוני המיקום של הפלאפון, ובחומרה אשר מסיקה על סטיית ראש המכשיר מהצפון הגיאוגרפי, לכן אין באפשרותנו להבטיח כי היא תעבוד טוב בכל מזג אוויר, ועל כל מכשיר. כלומר אנו משערים כי במכשירים שונים האפליקציה תעבוד באיכות שונה, וכי במזג אוויר מעונן או סוער הדיוק לא יהיה מיטבי. בנוסף כאשר האפליקציה מבקשת למצוא מיקום של שני אנשים שנמצאים בהפרש גבהים מסוים, או כאשר שניהם עומדים על צלע של הר או גבעה, נתוני המיקום גם כן לא יהיו מדויקים בהכרח.

לדוגמא: בטכניון קיבלנו תוצאות מדויקות יותר כאשר שני המשתמשים היו בשדרה שמתחת לדשא המרכזי מאשר כשאחד מהם היה בחצי הגובה של הדשא המרכזי.

כיוונים עתידיים

1. <u>שינוי מסך ה- AR -</u> לדעתנו יש מקום לשפר את האופן שבו מוצגת הקפסולה. ניתן לשפר את האופן שבו היא מתמזגת עם הסצנה ע"י שימוש בפיצ'רים חדשים של חבילות מציאות רבודה אשר דואגים למקם עצמים שאינם אמיתיים בצורה אמיתית יותר- כמו למשל פיצ'ר ה-Ground Plane של Vuforia.

- <u>ביטול הסיבוב-</u> ביטול סיבוב הקפסולה יעזור למשתמש להתרכז ביעד עצמו.
- <u>החלפת הקפסולה בעיגול צמוד למישור הקרקע-</u> הקפסולה שיצרנו מנסה לדמות שטח תלת מימדי על תמונה דו-מימדית. ניתן ליצור עיגול סביב היעד, באופן שיתמזג עם מישור הקרקע.
- <u>ביטול הצגת הקפסולה בטווח קרוב-</u> ניתן להסיר את הקפסולה מהמסך בטווח קרוב ליעד, ובמקום ליידע את ה"מחפש" שהחיפוש נגמר (בדומה להכרזת Waze "הגעת ליעד!").
- <u>הצגת החץ והקפסולה על אותו המסך-</u> כדי לחסוך מעבר בין המסכים ניתן לשים את המצפן והקפסולה על אותו המסך.

2. <u>אינטגרציה עם תוכנות אחרות-</u> לדעתנו נקודת החולשה העיקרית של האפליקציה כיום היא הפעולות שהמשתמש צריך לעשות כדי להתחיל את תהליך החיפוש. משימה עתידית יכולה להיות יצירת ממשק עם WhatsApp או כל תוכנה אחרת, אשר תגרום לתהליך החיפוש להיות יותר מהיר ועם פחות פעולות. הצעה לממשק: מי שרוצה שימצאו אותו יוצר חדר בעצמו עם מספר אנשים שברצונו לשתף איתם את מיקומו, ואז לוחץ על כפתור והשיתוף מתחיל אוטומטית.

3. <u>הוספת פונקציונאליות נוספת-</u> כרגע האפליקציה אינה מתחשבת ב"גובה", אף על פי שבפועל כן קיימים נתוני הגובה של כלל המשתמשים. ניתן למשל להוסיף הודעות שמתריעות על הפרשי גבהים מסוימים. בנוסף ניתן להוסיף התרעות על "שגיאות גדולות" של נתוני המיקום, או התחשבות בזמני התאורה של היום ובהתאם לכך לשנות את סט הצבעים של כלל הממשק.

4. <u>חיסכון סוללה-</u> ניתן לסגור את האפליקציה ברגע שהמטרה נמצאה ובכך לחסוך בחיי סוללה.

סיכום

לסיכום, אנו מודים לירון חונן אשר ליווה אותנו בתהליך מימוש האפליקציה. היה לנו חשוב לקדם רעיון שהוא שלנו, והעובדה שהתהליך קרה תחת מטריית המעבדה של ירון ובליוויו תרמה לנו מאד.

למדנו המון על Unity ועל האתגרים שבבניית אפליקציית מציאות רבודה עבור משתמשים שאינם בקיאים בניווטים.

הסמסטר נגמר אך כמובן שעוד ניתן לקדם ולפתח את האפליקציה שבנינו, וכפי שציינו במהלך הדו"ח- אנחנו בטוחים כי בעתיד יופץ מוצר אשר דומה במהותו למה שפיתחנו.

בהערכה רבה,

- מיה סאבו ולב טוניק. (חורף 2020)