הטכניון – מכון טכנולוגי לישראל

הפקולטה להנדסת חשמל

Laboratory of Networked Software Systems

**בית חכם עם רשת חברתית**

אביב תשע"ח

מגיש: אלכסיי חרומוב מנחה: אורן קלינצקי

תוכן עניינים

[1. מבוא 3](#_Toc509335649)

[2. אתגרים בפיתוח אפליקציות ל- VR 4](#_Toc509335650)

[3. SDKs 5](#_Toc509335651)

[DirectX11 5](#_Toc509335652)

[FFmpeg 5](#_Toc509335653)

[Intel Media SDK 5](#_Toc509335654)

[Oculus PC SDK 6](#_Toc509335655)

[4. מבט על 8](#_Toc509335656)

[ספריות סטטיות 8](#_Toc509335657)

[כלי פיתוח ופרופיל 8](#_Toc509335658)

[5. ארכיטקטורה 9](#_Toc509335659)

[ארגומנטים 9](#_Toc509335660)

[סקירה 9](#_Toc509335661)

[אתחול 10](#_Toc509335662)

[ריצה 11](#_Toc509335663)

[ספרה 12](#_Toc509335664)

[ממשק CHWDevice 15](#_Toc509335665)

[מחלקת CD3D11DeviceOVR 15](#_Toc509335666)

[מחלקת CDecodeD3DRender 16](#_Toc509335667)

[תרשים מחלקות 17](#_Toc509335668)

[אלגוריתם להמרת קצב ריענון 17](#_Toc509335669)

[6. פרופיל ריצה 20](#_Toc509335670)

[Visual Studio Performance Profiler 20](#_Toc509335671)

[Intel VTune Amplifier 21](#_Toc509335672)

[7. סיכום 21](#_Toc509335673)

[8. ביבליוגרפיה 22](#_Toc509335674)

[9. נספחים 22](#_Toc509335675)

[בדיקת האלגוריתם 22](#_Toc509335676)

[10. רשימת איורים 23](#_Toc509335677)

1. מבוא

העולם סביבנו כבר השתנה, הטכנולוגיה כבר נמצאת בבית של כל אחד ואחת מאיתנו, כולנו נושאים אותה בכיס, ואפילו עונדים אותה במקום שעון קוורץ. אז למה לא להכריח אותה להציג לנו מראש את מה שאנחנו אוהבים?

ניקח דוגמה: אדם קם בבוקר, רוצה לראות מה חדש בעולם החדשות, הספורט או הסדרות האהובות עליו, הוא כבר עשה "לייק" לעמודים האהובים שלו בפייסבוק, נשאר רק שהמחשב יציג לו את העדכונים החדשים מתוכם!

כאן הבית החכם נכנס לפעולה: נעמיד מחשב 3raspberry pi עם מצלמה ומסך מחוברים אליה, המצלמה תקלוט ותזהה את האדם העומד מולה, המסך יציג את התוכן האהוב עליו מאחד עמודי פייסבוק.

כעת נניח שאדם קורא תוכן והוא רוצה לגלול מטה או שהוא דווקא רוצה להחליף עמוד, למה "להתאמץ" בלחיצת כפתורי מקלדת \ עכבר, אם אפשר פשוט לעשות תנועות ידיים באוויר?

נכון! המערכת תזהה את תנועות הידיים ותפעל לפיהם.

1. אתגרים בפיתוח אפליקצייה חכמה

בפיתוח האפליקציה עמדו לפנינו אתגרים רבים. ראשית התממשקות עם SDKs שונים (יפורטו בהמשך), עמידה בדרישות אבטחה של פייסבוק ולבסוף שימוש במחשב raspberry pi3 לתפעול האפליקציה.

* האתגר הכי גדול בפיתוח התגלה דווקא לאחר ניסוח הפרוייקט:

מתוך ויקיפדיה:

**קיימברידג' אנליטיקה** (ב[אנגלית](https://he.wikipedia.org/wiki/%D7%90%D7%A0%D7%92%D7%9C%D7%99%D7%AA" \o "אנגלית): **Cambridge Analytica**) הייתה [חברה פרטית](https://he.wikipedia.org/wiki/%D7%97%D7%91%D7%A8%D7%94_%D7%A4%D7%A8%D7%98%D7%99%D7%AA) [בריטית](https://he.wikipedia.org/wiki/%D7%91%D7%A8%D7%99%D7%98%D7%A0%D7%99%D7%94) ששילבה [כריית נתונים](https://he.wikipedia.org/wiki/%D7%9B%D7%A8%D7%99%D7%99%D7%AA_%D7%A0%D7%AA%D7%95%D7%A0%D7%99%D7%9D), מסחר בנתונים, [ניתוח מידע](https://he.wikipedia.org/wiki/%D7%A0%D7%99%D7%AA%D7%95%D7%97_%D7%9E%D7%99%D7%93%D7%A2) ואסטרטגיה תקשורתית לצורך השפעה על תהליכי [בחירות](https://he.wikipedia.org/wiki/%D7%91%D7%97%D7%99%D7%A8%D7%95%D7%AA) במדינות ברחבי העולם[...] בשנת [2016](https://he.wikipedia.org/wiki/2016) הייתה החברה מעורבת במסע [הבחירות לנשיאות ארצות הברית](https://he.wikipedia.org/wiki/%D7%94%D7%91%D7%97%D7%99%D7%A8%D7%95%D7%AA_%D7%9C%D7%A0%D7%A9%D7%99%D7%90%D7%95%D7%AA_%D7%90%D7%A8%D7%A6%D7%95%D7%AA_%D7%94%D7%91%D7%A8%D7%99%D7%AA_2016) של [דונלד טראמפ](https://he.wikipedia.org/wiki/%D7%93%D7%95%D7%A0%D7%9C%D7%93_%D7%98%D7%A8%D7%90%D7%9E%D7%A4) ובקמפיין ה[ברקזיט](https://he.wikipedia.org/wiki/%D7%91%D7%A8%D7%A7%D7%96%D7%99%D7%98) ליציאת [בריטניה](https://he.wikipedia.org/wiki/%D7%91%D7%A8%D7%99%D7%98%D7%A0%D7%99%D7%94) מ[האיחוד האירופי](https://he.wikipedia.org/wiki/%D7%94%D7%90%D7%99%D7%97%D7%95%D7%93_%D7%94%D7%90%D7%99%D7%A8%D7%95%D7%A4%D7%99" \o "האיחוד האירופי). תפקודה של קיימברידג' אנליטיקה בקמפיינים הללו הוא נושא לחקירות פליליות מתמשכות בשתי המדינות, בעיקר בנוגע לשיטות בהן היא נוקטת לצורך מיקוד בוחרים.]..] ב-[17 במרץ](https://he.wikipedia.org/wiki/17_%D7%91%D7%9E%D7%A8%D7%A5) [2018](https://he.wikipedia.org/wiki/2018), דיווחו העיתונים "[ניו יורק טיימס](https://he.wikipedia.org/wiki/%D7%A0%D7%99%D7%95_%D7%99%D7%95%D7%A8%D7%A7_%D7%98%D7%99%D7%99%D7%9E%D7%A1)" ו"[האובזרבר](https://he.wikipedia.org/wiki/%D7%94%D7%90%D7%95%D7%91%D7%96%D7%A8%D7%91%D7%A8)" שקיימברידג' אנליטיקה עשתה שימוש עסקי במידע אישי מ[פייסבוק](https://he.wikipedia.org/wiki/%D7%A4%D7%99%D7%99%D7%A1%D7%91%D7%95%D7%A7), שמלכתחילה נאסף על ידי חוקר חיצוני למטרות אקדמיות. בתגובה, אסרה פייסבוק על קיימברידג' אנליטיקה לפרסם בפייסבוק.

האירועים הנ"ל רגמו לפייסבוק לבצע שינוי משמעותי באבטחה. החל מאפריל לא ניתן לתת הרשאות publish\_actions לאפליקציות – הרשאה שהיה בה צורך להעלאת תמונה לקיר של המשתמש הראשי – כלומר הבית.

* אתגר נוסף שעמד לפנינו היה שליפת תמונה מתוך streaming של מצלמה והעברתה לשרת לצורך זיהוי. פתרון שלבסוף לא נלקח משום שלא היה יעיל: לעשות THREAD בצד שרת של המצמלה, ופעם בחצי שנייה לעצוןר אותו, לעשות שליפת תמונה ואז להמשיך את הSTREAMING. הפתרון כמובן לא טוב, אין סיבה לעצור את הזרמת המצלמה בשביל לקחת תמונה, זה עלול לגרום לתקיעות ולפעולות מיותרות של התוכנה.

הפתרון שכן נלקח לבסוף הוא שימוש במצלמה מצד הלקוח, שירות הSTREMING מבוצע ישירות בצד לקוח ע"י JAVASCRIPT, פעם בחמי שנייה הסקריפט ייקח SCREENSHOT מתוך המצלמה, בלי לעצור את הזרמת הנתונים למסך, ויישלח את הלינק הזמני של התמונה לצד שרת. השרת ייגש ללינק הזמני, ישמור את התמונה ולאחר מכן יעבד אותה – בין אם מדובר בזיהוי פנים ובין אם מדובר בזיהוי תנועות.

* האתגר האחרון שעמד לפנינו היה זיהוי תנועות ידיים. על מנת לזהות את התנועות השתמשנו באלגוריתם הבא: לקחנו 2 תמונות עוקבות בהפרש של חצי שנייה ועשינו על שתיהן אתץ הפעולות הבאות:
* תמונה מקורית:



* הפוך את התמונה לגווני שחור – לבן:



* טשטש את התמונה מעט על מנת להמנע מרעשים:



* קיטוב תבעי התמונה לשחור אם הצבע בין 0 ל100, וללבן אם הצבע בין 101-255:



* וכעת ניתן לחשב את ממוצע הצבע הלבן בתמונה, למשל: [285, 261].
* נזכור שכרגע יש בידינו 2 תמונות, במרווח זמן של חצי שנייה, ונראה לאיזה כיוון זז ממוצע הצבע הלבן בתמונה השנייה, לפי כיוון התזוזה המקסימלי נדע אם תנועת היד הייתה בכיוון מעלה \ מטה \ ימינה \ שמאלה.

1. SDKs

להשלים

1. מבט על

האפליקציה עובדת בצורת תכן סטנדרטית של צד לקוח וצד שרת:

Response uploaded picture ID  
+ Recognized face

Ask for picture upload   
+ face recognition

Non-stop streaming from camera to client-side

צד שרת

צד לקוח

Facebook



כל חצי שנייה: צלם תמונה ושלח בקשה לצד שרת.  
אם שנייה עגולה שלח בקשה לזיהוי פנים, אחרת שלח בקשה לזיהוי תנועה.  
פעל לפי תשובת השרת שהתקבלה: דפדוף אנכי, שינוי עמוד או שינוי זיהוי.

אם הגיע בקשת זיהוי פנים: העלאה את התמונה הנוכחית לפייסבוק, ובקש זיהוי.  
אם הגיע בקשת זיהוי תנועה, השווה את 2 התמונות האחרונות. בכל מקרה שלח תשובה ללקוח.

1. ארכיטקטורה ומימוש

צד לקוח:

עבור מימוש כל אחת מהבקשות לFLASK, עם JSON QUERY נשתמש במבנה הבא:

$.getJSON(  
 {  
 **url**: **<url which describes the wanted request>**,  
 **data**: {},  
 success: **function** (result) {  
 **if** (result.**status** == **"new\_fb\_page"**) {  
 $(**".fb-page"**).attr(**'data-href'**, result.next\_url);  
 ***document***.getElementById(**"page\_name\_span"**).**textContent** = result.page\_name;  
 }  
 }  
 });

**function** *get\_next\_page*(direction)

* הפונקציה נקראת כאשר המשתמש הנוכחי דפדף ימינה או שמאלה, הצד-לקוח יבקש מהשרת את העמוד הבא (או הקודם) מרשימת העמודים האהובים על המשתמש.

**function** *get\_person\_name*()

* הפונקציה נקראת כל שניה עגולה, ומעבירה לצד שרת את התמונה הנוכחית במצלמה, וזאת עבור זיהוי פנים של המשתמש.

**function** *get\_gesture*()

* הפונקציה נקראת כל חצי שניה מהצד-לקוח לצד שרת, ומבקשת ממנו זיהוי תנועת יד לפי 2 תמונות אחרונות שצולמו.

פונקציית הזרמת תמונה ללא הפסקה (STREAMING):

***navigator***.**mediaDevices**.getUserMedia(***constraints***).then(*handleSuccess*).catch(*handleError*);  
**function** *handleSuccess*(stream) {  
 ***video***.**srcObject** = stream;  
}  
**function** *handleError*(error) {  
 ***console***.error(**'Error: '**, error);  
}

פונקציית לקיחת SCREENSHOT רגעי מהמצלמה:

**function** *get\_screenshot*()  
{  
 ***canvas***.**width** = ***video***.**videoWidth**;  
 ***canvas***.**height** = ***video***.**videoHeight**;  
 ***canvas***.getContext(**'2d'**).drawImage(***video***, 0, 0);  
 ***img***.**src** = ***canvas***.toDataURL(**'image/png'**);  
}

פרופיל ריצה

בפרק זה נציג את ממצאי הפרופיל שאספנו עבור ריצה אופיינית של התכנית. לצורך האיסוף השתמשנו בשני כלים: Performance Profiler המובנה ב Visual Studio 2017 ו VTune Amplifier של אינטל, כאשר היתרון של הכלי האחרון הוא היכולת למדוד משימות ספציפיות.

## Visual Studio Performance Profiler

פרופיל אופייני שנאסף עם כלי זה מוצג בגרף הבא

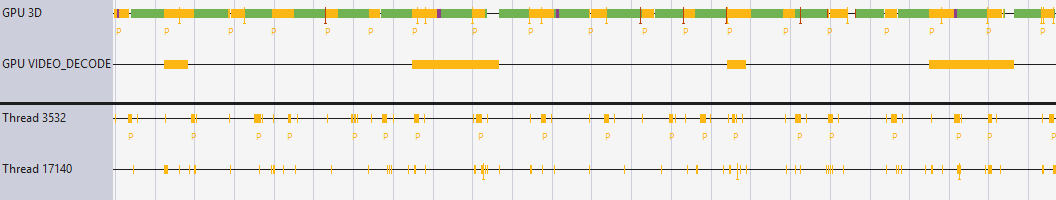


Figure 17- Visual Studio Performance Profiler

הצבע הצהוב מסמן את חלק הזמן שבו הנגן שלנו עבד על ה- GPU ואילו החלק הירוק מסמן את חלק הזמן שבו ה- Runtime של Oculus עבד על ה- GPU. האות p מציינת קריאה ל Present().

חלק הזמן שבו פועלת כל קריאה ל- GPU מסוכמת בסכמה הבאה:

Eye 1

Eye 2

Figure 18 - Rendering loop time distribution

עיבוד הנתונים הראה שמרבית הזמן שלוקח לרנדר פריים נצרך על ידי ה Runtime של Oculus לשם רינדור העיוות הנדרש מסביב לכל עין וזהו צוואר הבקבוק המשמעותי ביותר בהליך הרינדור.

## Intel VTune Amplifier

כלי זה ניתן לתכנות על מנת למדוד את אורכן של משימות ספציפיות לאורך ריצת הנגן. לכן ביקשנו למדוד את זמן ביצוע האתחול, את הזמן שלוקח לקרוא מידע חדש לתוך ה Bitstream ואת הפעולות שמתבצעות בלולאת הרינדור על ה- GPU. פרופיל אופייני שנאסף עם כלי זה מוצג בגרף הבא

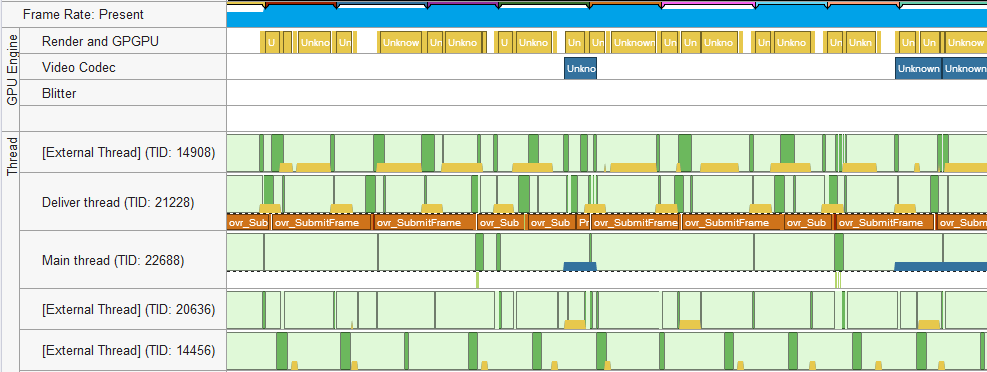


Figure 19 - Intel VTune Amplifier

הסימניות בשורה העליונה מציינות קריאה ל Present(), הצבע הצהוב מציין פעולת רינדור שמתבצעת על ה- GPU, הצבע הכחול מציין פעולת פענוח שמתבצעת על ה- GPU. בצד שמאל מצוין ליד כל חוט מה שמו (Main, Deliver) או שמא הוא שייך לתהליך חיצוני (ה- Runtime).

גם בדיאגרמה זו אפשר לראות שמרבית הזמן שלוקח לרנדר פריים נצרך על ידי ה Runtime כפי שראינו עם הכלי של Microsoft. מצאנו גם שפעולת האתחול של הנגן מסתיימת תוך כ- 0.2 שניות, וכן שמילוי ה Bitstream אורך פחות מ- 1 ms, כלומר פעולות אלו מתבצעות ביעילות ולא מהוות צוואר בקבוק. כמו כן אפשר לראות שהקריאה ל SubmitFrame חוזרת בסמוך לפעולת הריענון של הקסדה והיא זו שמהווה את צוואר הבקבוק המשמעותי של הנגן. בין הקריאות לפונקציה זו ניתן גם למצוא את יתר הפקודות שמדדנו כמו Present, CopyResource, Clear Buffers, Draw וראינו שהן מתבצעות מהר מאוד יחסית ל SubmitFrame.

1. סיכום

בפרויקט זה פיתחנו נגן VR שיכול לנגן סרטים שצולמו בטכנולוגיית 360 מעלות. הנגן יודע לקרוא את כל הפורמטים הסטנדרטיים הודות לשימוש ב FFmpeg, לשם המחשת יכולות הכנסנו תמיכה במשקפי Oculus Rift אך המבנה הגמיש של הנגן מאפשר להכניס בקלות תמיכה במשקפיים של כל חברה אחרת. הנגן מריץ שני חוטים בתצורת יצרן-צרכן שפועלים ביעילות ומאותתים זה לזה. על מנת לשפר את חווית המשתמש, כתבנו אלגוריתם להמרת קצב ריענון והוכחנו שהוא אופטימלי. לבסוף, אספנו פרופיל אופייני של הנגן בעזרת שני כלים שונים ומצאנו שה- Runtime של Oculus מהווה את צוואר הבקבוק המשמעותי ביותר.

1. ביבליוגרפיה

Intel. (n.d.). *Intel Media Developers Guide.*

Intel. (n.d.). *Media SDK Manual.*

Luna, F. D. (2012). *Introduction to 3D Game Programming with DirectX 11.*

METTLE. (n.d.). Retrieved from https://www.mettle.com/360vr-master-series-free-360-downloads-page/

Microsoft. (n.d.). *DirectXTK Wiki.* Retrieved from Github: https://github.com/Microsoft/DirectXTK/wiki

Oculus. (n.d.). *Introduction to the PC SDK*. Retrieved from https://developer.oculus.com/documentation/pcsdk/latest/concepts/pcsdk-intro/

1. נספחים

## בדיקת האלגוריתם

בעזרת תכנית זו בדקנו את אופן פעולת האלגוריתם להמרת קצב הריענון

#include **<iostream>**#define **HMD\_REFRESH\_RATE** 75  
#define **SRC\_FPS** 30  
  
**void** loop(**int** p, **int** q)  
{  
 **int** mod\_p, mod\_q = 0;  
  
 **int** i = 150;  
 **while**(i--)  
 {  
 mod\_p = 1 % p;  
 std::cout << **"in "**;  
 **while** (mod\_p) {  
 mod\_p = (mod\_p + 1) % p;  
 std::cout << **"loop "**;  
 }  
 **do** {  
 mod\_q = (mod\_q + q) % (HMD\_REFRESH\_RATE - (p - 1) \* SRC\_FPS);  
 std::cout << **"loop"** << **" "**;  
 } **while** (mod\_q / SRC\_FPS);  
 std::cout << **"out"** << std::endl;  
 }  
}  
  
**int** main() {  
 loop(HMD\_REFRESH\_RATE / SRC\_FPS, HMD\_REFRESH\_RATE % SRC\_FPS);  
 **return** 0;  
}

1. רשימת איורים

[Figure 1- Oculus Rift 3](#_Toc509335725)

[Figure 2- Standard containers 5](#_Toc509335726)

[Figure 3 - Ideal VR Pipeline 6](#_Toc509335727)

[Figure 4 - Dropped Frame 6](#_Toc509335728)

[Figure 5 - Warp Miss 7](#_Toc509335729)

[Figure 6- Libraries Dependency Graph 8](#_Toc509335730)

[Figure 7- Player overview 9](#_Toc509335731)

[Figure 8 - CDecodingPipeline::Init() 10](#_Toc509335732)

[Figure 9 - CDecodingPipeline::Run() 11](#_Toc509335733)

[Figure 10 - Sequence Diagram 12](#_Toc509335734)

[Figure 11 - Sphere tessellation=3 13](#_Toc509335735)

[Figure 12 - Sphere tessellation=4 13](#_Toc509335736)

[Figure 13 - Sphere tessellation=5 14](#_Toc509335737)

[Figure 14 - Sphere tessellation=6 14](#_Toc509335738)

[Figure 15 - Output example 16](#_Toc509335739)

[Figure 16 - Class Diagram 17](#_Toc509335740)

[Figure 17- Visual Studio Performance Profiler 20](#_Toc509335741)

[Figure 18 - Rendering loop time distribution 20](#_Toc509335742)

[Figure 19 - Intel VTune Amplifier 21](#_Toc509335743)