



Visual Looming VR

מגישים: ענבר דונג ועלאא חיר

מנחים: דניאל רביב, בועז שטרנפלד, ירון חונן ואלון זבירין

תאריך: 23.7.2017



תוכן עניינים

2	מבוא ותקציר
5	תיאור המערכת
8	תיאור הביצוע
14	תוצאות
15	מסקנות
16	המלצות
16	מקורות ספרותיים
16	נספחים

מבוא ותקציר

מטרת הפרויקט הינה לממש אלגוריתם שיאפשר לגוף מסוים להימנע מהתנגשויות במכשולים בסביבות לא מוכרות לגוף ולבנות סימולציה ב Unity כדי להמחיש את האלגוריתם. Looming הוא ערך שאפשר לחשב עבור מכשול ופירושו כמות הסכנה – ככל שהגוף מתקרב למכשול כמות הסכנה שלו עולה, ועל הגוף להימנע ממגע עם המכשולים בעלי ערכי ה- looming הגבוהים יותר בכל רגע ורגע.

אנחנו בפרויקט מימשנו את האלגוריתם בשתי שיטות:

1. שיטה מדויקת שבה אנו משתמשים במרחקים מהמצלמה למכשולים. בשיטה זו ידועים ערכי ה- looming לכל פיקסל בתמונת המתקבלות מהמצלמה.
2. שיטה של ראייה ממוחשבת שבה איננו מודדים מרחקים אלא משתמשים אך ורק במידע שניתן להשיג מרצף תמונות מהמצלמה. וליתר דיוק משתמשים בערכי צפיפות הטקסטורה מהתמונות.



שיטה 1 – כדי ליצור את התמונה שבצד שמאל למעלה היה צריך להשתמש במרחקים מהמצלמה לכל פיקסל בתמונה שמתקבלת מהמצלמה של הרכב.



שיטה 2 – שימוש בצפיפות הטקסטורה לחישוב looming. כשהרכב מתקרב למכשול (במקרה הזה בעל טקסטורה של עלים) הצבע הופך בהדרגה מירוק לאדום.

לאחר שחישבנו Looming בשתי השיטות אנחנו השווינו ובדקנו עד כמה שיטה 2 מדויקת ביחס לשיטה 1 בחישוב ה- looming .

בנוסף מימשנו ויזואליזציות שונות שמתארות את ערכי ה- looming בעזרת צבעים בגוונים של אדום עד ירוק, כאשר ירוק מסמל אזורים בטוחים (looming נמוך) ואדום מסמל אזורים מסוכנים (looming גבוה). אחת מהויזואליזציות היא מפת looming ונפרט על עוד ויזואליזציות בהמשך.



מפת looming. שימו לב כי העץ צבוע באדום במפה וזה באמת המכשול הקרוב ביותר למכונית.

לאחר מכן מימשנו עבור הגוף אלגוריתם לקבלת החלטות אשר מחשב לאן לפנות לפני התנגשות (ברגע שה- looming גבוה והצבע מספיק אדום בכיוון ההתקדמות). ההחלטות מבוססות על ערכי looming שחושבו בשיטה 1.

לבסוף השתמשנו ב Oculus Rift בשביל לממש חווית מציאות מדומה (VR) של נסיעה במכונית ללא נהג. הוספנו את האפשרויות הבאות:

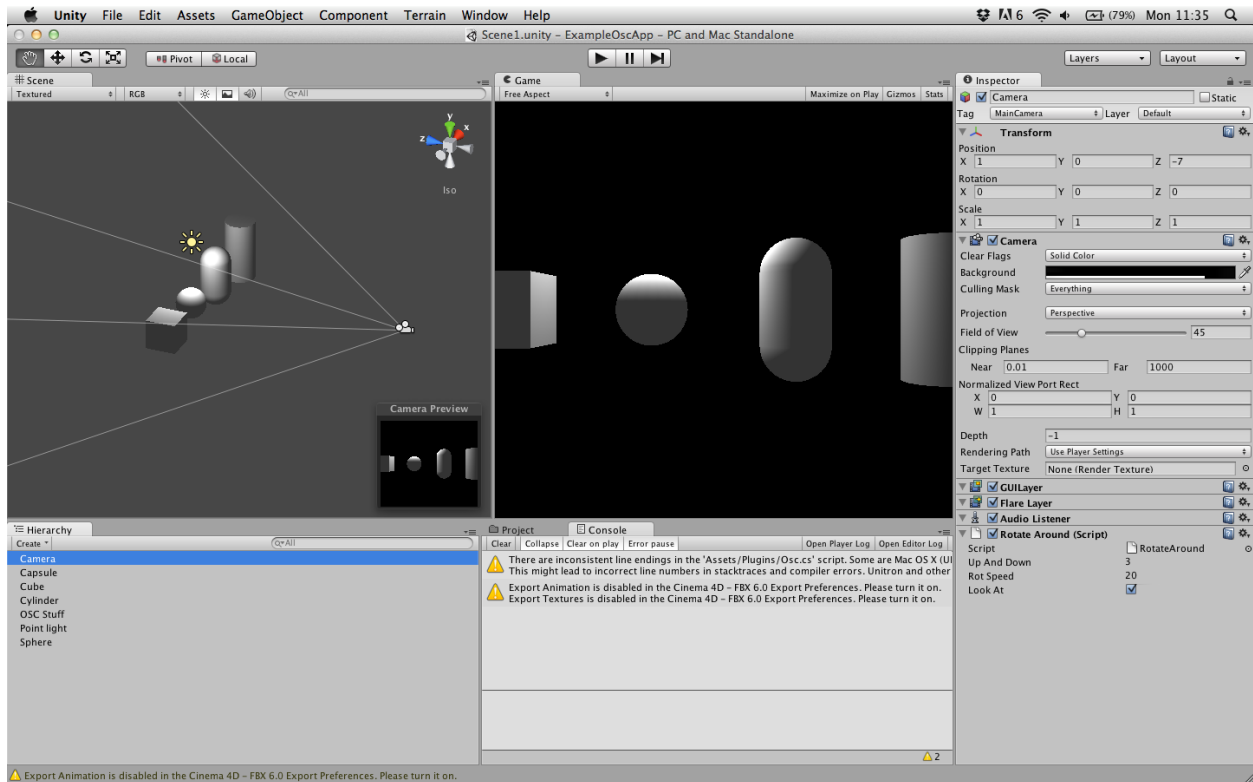
1. להפעיל ויזואליזציות שונות
2. לשנות את מהירות המכונית
3. לעוף ולהסתכל על המכונית נעה מכל זווית שהי
4. לחזור למכונית בלחיצת כפתור



כאן מודגמת האפשרות לשנות את המהירות ולצפות במפת ה looming ב VR.

תיאור המערכת

בפרויקט שלנו השתמשנו בשתי מערכות: Unity ו-Oculus Rift. Unity הוא מנוע גרפי חוצה פלטפורמות ומשמש לפיתוח משחקי וידאו וסימולציות למחשב, קונסולות, טלפונים חכמים ואתרי אינטרנט. אפשר באמצעותו לבנות סביבה תלת ממדית ולתכנת סקרिפטים עבור גופים תלת מימדיים אשר יכתיבו את התנהגותם וכיצד הם נעים במרחב.



דוגמה פשוטה ב Unity - בסצנה מצד שמאל ניתן לראות שיש 4 אובייקטים, מצלמה ומקור אור. בצד ימין רואים את התמונה שנוצרת ע"י המצלמה.

Oculus Rift היא מערכת מציאות מדומה המאפשרת למשתמש לחוות את העולם התלת מימדי כאילו הוא נמצא בתוכו. המערכת מכילה משקפיים, 2 שלטים (אחד לכל יד) ו-2 חיישנים אשר עוקבים אחרי המשקפיים והשלטים ונותנים לנו לדעת איפה נמצאים הראש והידיים של המשתמש וגם את האוריינטציה שלהם בעולם התלת מימדי.



Oculus Rift - משקפיים + 2 חיישנים + 2 שלטים

במהלך הפרויקט שלנו Unity איפשר לנו:

1. לבנות סביבה תלת ממדית המכילה מודלים כגון בתים, רכבים, כדורים וכו'.
2. סביבה דינאמית כפונקציה של הזמן. למשל – תנועת הרכב נשלטת בזמן אמת ע"י סקריפט המצורף למצלמה שעל הרכב. הרכב כבן של המצלמה בהיררכיית האובייקטים יורש מהמצלמה את התנועה שלה וכך נע ביחד איתה.
3. ראייה של הסביבה מזוויות שונות. יכולנו להזיז את המצלמה לכל מקום ולכוון אותה בכל כיוון. בפרויקט שלנו יש 2 מצלמות – מצלמה אחת מצורפת לרכב ונותנת רצף של תמונות לאלגוריתם ה Looming. ומצלמה שנייה מצלמת כך שהמשתמש יוכל לראות מזוויות שונות את העולם, למשל מבט של ציפור מעל הרכב.
4. לשנות חומרים שמהם האובייקטים בנויים. למשל לבחור עבורם צבעים, טקסטורות ו shaders. בעזרת shader ניתן לקבוע בקוד שרץ במקביל לכל פיקסל או קודקוד ב GPU את תכונותיהם השונות. למשל לכל פיקסל אפשר לקבוע צבע.
5. אפשרות לעבד אינפורמציה של תלת ממד ודו ממד בסקריפטים המצורפים לאובייקטים. למשל אלגוריתם קבלת החלטות לרכב השתמש בתמונות שהתקבלו מהמצלמה המצורפת לרכב לאחר שעברו עיבוד כדי לקבל החלטה בכל רגע – האם לזוז שמאלה, ימינה או להמשיך ישר.

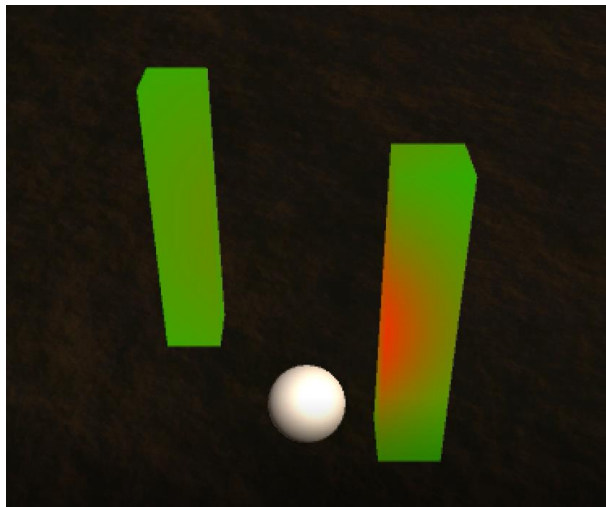
1. להוסיף מצלמת VR שזזה באותו האופן שבו הראש זז עם המשקפיים. למצלמה הזו אפשר כמו לכל אובייקט ב Unity להוסיף סקריפטים נוספים ולגרום לה לרשת תנועה של אובייקט אחר.
2. היכולת לעוף מומשה בסקריפט שצורף למצלמת VR. כשהתוכנה מתחילה לרוץ מצלמת ה VR נעה יחד עם המצלמה של הרכב, וכך המשתמש רואה את העולם מנקודת מבט של נוסע ברכב. אם נלחץ על כפתור התעופה בשלט הימני, מצלמת ה VR משתחררת מהרכב ומיקומה מתחיל להשתנות בהתאם לקלט מהמשתמש – כיוון השלט הימני כאשר הכפתור לחוץ.
3. היכולת לחזור לרכב בלחיצת כפתור – מצלמת ה VR חוזרת לאותו המקום שבו מצלמת הרכב נמצאת.
4. כפתור הגדרות פותח GUI שבו ניתן לראות את מפת ה Looming, לשנות את מהירות הרכב, להדליק ולכבות כדורים.

תיאור הביצוע

מימוש בשיטת המרחקים

בהתחלה מימשנו את האלגוריתם בשיטה 1, שבה אנו מודדים מרחקים מכל פיקסל של מכשול למצלמה. נסמן מרחק זה ב- R . ידוע לנו וקטור המהירות והמיקום של המצלמה בכל רגע ורגע. ב fragment shader אנו מחשבים את הצבע המתאים לערך ה- $looming$ של הפיקסל. ככל שערך ה- $looming$ של הפיקסל יותר גבוה, הפיקסל יותר אדום. להלן הקוד שמחשב את ערך ה- $looming$ וגם את הצבע המתאים:

```
// fragment shader
float4 frag (v2f i) : SV_Target
{
    float3 R = i.worldPos - _CamPosition; //range
    float L = dot(_CamVelocity,R) / dot(R,R); //looming
    float4 col = lerp(float4(0.0f,1.0f,0.0f,1.0f), float4(1.0f,0.0f,0.0f,1.0f), L);
    return col;
}
```



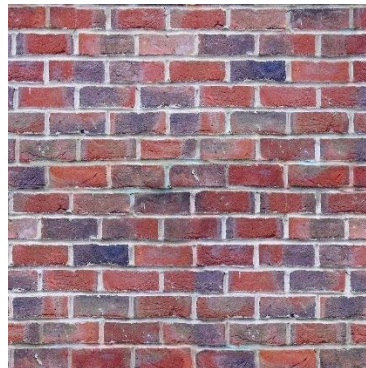
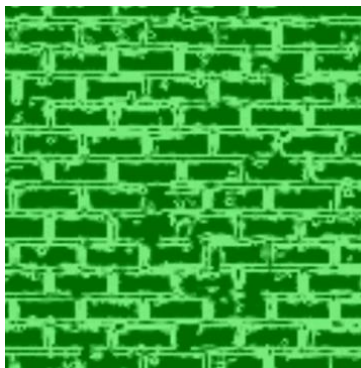
ה shader צובע את המכשולים בצבעים המתאימים לערכי ה- $looming$ שלהם

מימוש בשיטת צפיפות הטקסטורה

לאחר מכן התחלנו לממש את האלגוריתם בשיטה 2 המשתמשת בצפיפות הטקסטורה של המכשול. ככל שהמצלמה מתקרבת לטקסטורה עם הזמן, היא רואה את הטקסטורה הזו מתרחבת והופכת לפחות צפופה, ומכך אפשר להסיק שערך ה looming של המכשול עולה עם ההתקרבות לטקסטורה. הקלט לאלגוריתם הוא רצף של תמונות מהמצלמה של הרכב. ואנו רוצים לחשב את צפיפות הטקסטורה לכל תמונה וכיצד משתנה הצפיפות של הטקסטורה כפונקציה של הזמן. את ה looming ניתן לחשב לפי הנוסחה הבאה כאשר γ היא צפיפות הטקסטורה.

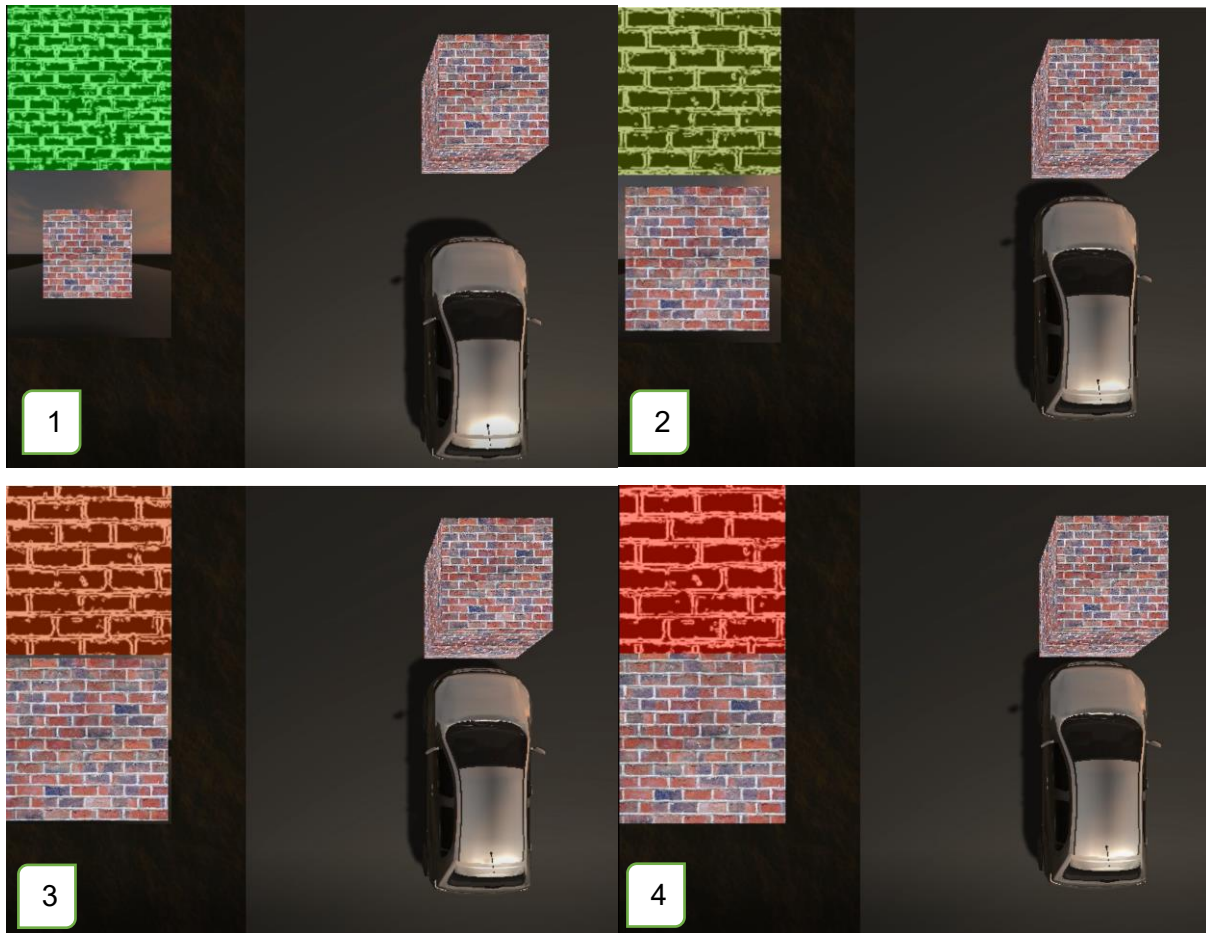
$$L = \frac{dy}{dt} \gamma$$

לפני חישוב של צפיפות הטקסטורה עשינו עיבוד לכל תמונה- הפעלנו את Sobel Filter כדי לקבל תמונה בשחור לבן כך שהקצוות של אלמנטים בתמונה מסומנים בלבן (עוזר להבדלה בין האלמנטים השונים בטקסטורה), ולאחר מכן גם מחקנו מהתמונה רעשים לבנים שהם קטנים ומיותרים לחישוב צפיפות הטקסטורה. את צפיפות הטקסטורה חישבנו ע"י ספירה של כמות השינויים משחור ללבן ולהיפך כאשר סורקים את התמונה מלמעלה למטה, ומשמאל לימין והממוצע שלהם. כדי לחשב את הנגזרות בכל פעם היינו צריכים לקרב אינטרוול מסוים של מדידות קודמות של צפיפויות טקסטורה לפולינום מסדר ראשון בגלל רעשים במדידות. לאחר שקיבלנו ערך מקורב של looming לכל תמונה חישבנו את הצבע המתאים ושילבנו אותו על גבי התמונה המעובדת.



מימין – לפני עיבוד, משמאל – אחרי עיבוד + הצבע המתאים משולב על גבי הטקסטורה

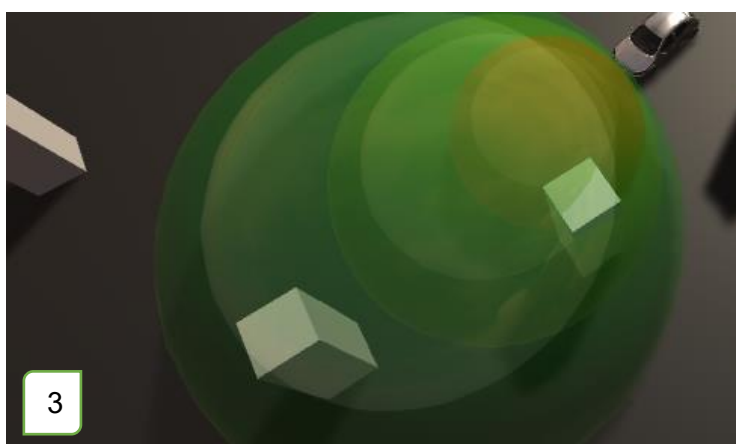
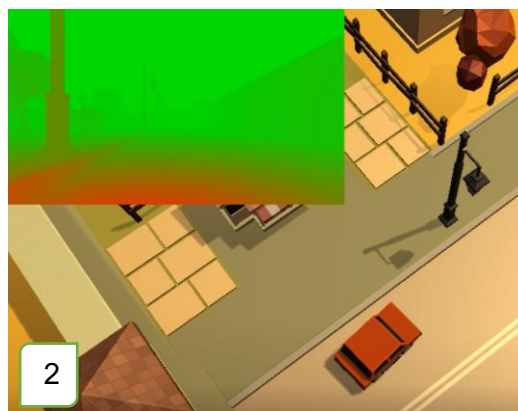
להלן רצף תמונות שמראה שככל שהרכב מתקרב לטקסטורה, הטקסטורה נעשית פחות צפופה והצבע הופך לאדום יותר באופן אקספוננציאלי.



ויזואליזציה

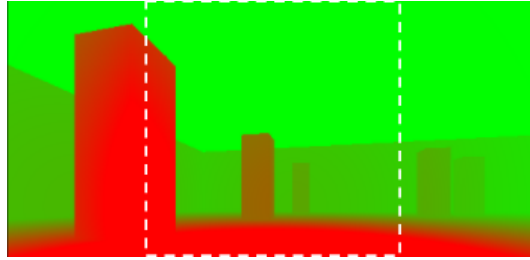
את הויזואליזציה של ה looming בסימולציה מימשנו ב-3 דרכים:

1. צביעה של המכשולים עצמם
2. צביעה של התמונות המתקבלות מהמצלמה
3. כדורים שגדלים ככל שהמהירות של הרכב עולה וקטנים כשהיא יורדת. מכשולים שבתוך הכדורים הפנימיים והאדומים יותר הם יותר מסוכנים ובעלי looming גבוה יותר. וככל שהמכונית יותר מהירה היא צריכה להיזהר גם מאובייקטים רחוקים יותר.



קבלת החלטות

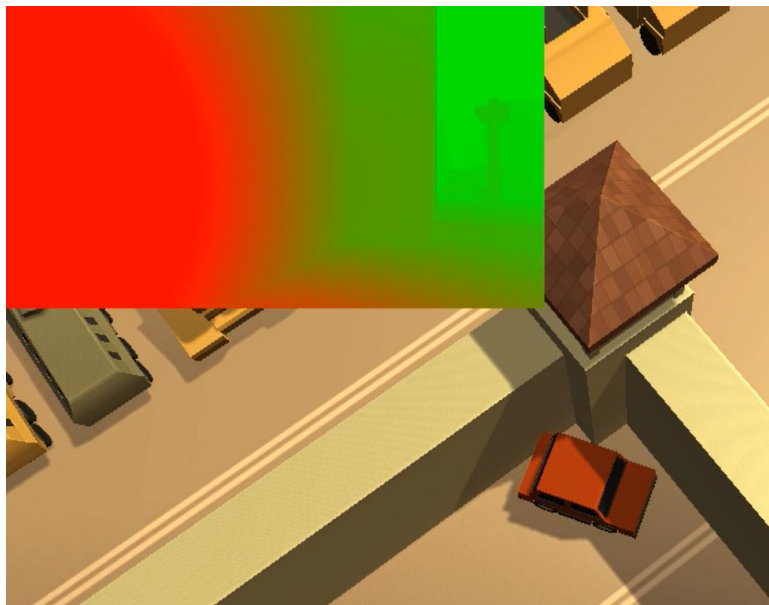
בכל רגע על המכונית לקבל החלטה – האם להמשיך ישר, שמאלה או ימינה.
 בשביל לקבל החלטות השתמשנו בתמונות המתקבלות מהויזואליזציה השנייה.
 אנחנו סורקים את אמצע התמונה בכל פעם ומחשבים את אחוז האדום בתמונה.



מחשבים באזור המקווקו את אחוז האדום

אם הוא מתחת לסף תחתון מסוים, זה אומר שאין סכנה מקדימה ואפשר להתקדם ישר. אם מעל הסף, צריך לפנות. ההחלטה אם לפנות שמאלה או ימינה מתקבלת לאחר סריקה של חצי שמאלי וחצי ימני של התמונה.

אם בחצי השמאלי יש אחוז של אדום גדול יותר מבחצי הימני, הרכב יפנה ימינה ולהיפך. לדוגמא בתמונה הבאה הרכב פונה ימינה כי בחצי השמאלי יש יותר אדום כי יש שם קיר.



הרכב פונה ימינה

מציאות מדומה

לבסוף השתמשנו ב Oculus Rift בשביל לתת למשתמש "להיכנס" לסימולציה שלנו ולחוות איך זה מרגיש להיות במכונית הנוסעת באופן אוטונומי. השתמשנו בכפתורים שבשלטי ה- touch בשביל האפשרויות הבאות:

1. תעופה
2. חזרה לרכב
3. הגדרות

בשביל לממש תעופה חשבנו על איך שסופרמן עף – הוא מושיט יד ימין ולאן שהיא מכוונת, לשם הוא עף. כך גם אצלנו, כאשר כפתור התעופה לחוץ, מצלמת ה VR נעה לכיוון השלט הימני שמחזיקה היד הימנית.

נתונים לנו ע"י Oculus וקטורי המיקום של המשקפיים והשלטים. לכן אנחנו יכולים בקלות לחשב את הכיוון הזה ע"י חיסור וקטור מיקום המשקפיים מוקטור מיקום השלט הימני.

בשביל לממש חזרה לרכב בלחיצת כפתור משנים את מיקום מצלמת ה-VR למיקומה של מצלמת הרכב.

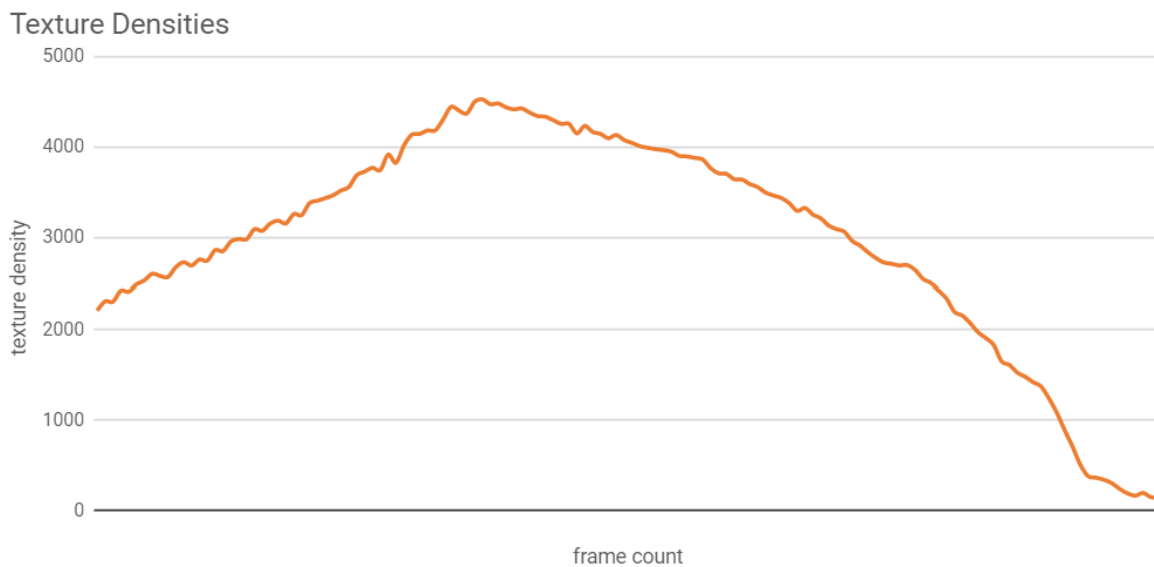
בהגדרות ניתן לצפות במפת ה looming, להדליק ולכבות כדורים עם checkbox, ולשנות מהירות עם slider. Checkbox ו- slider קיבלנו מספרייה מאוד מושקעת שנקראת Hover UI Kit (ראו נספח 1).

תוצאות

להלן הגרף שמראה את שינוי צפיפות הטקסטורה כפונקציה של הזמן עבור הטקסטורה הבאה. בדקנו עם עוד טקסטורות והגרפים שמתקבלים הם דומים.

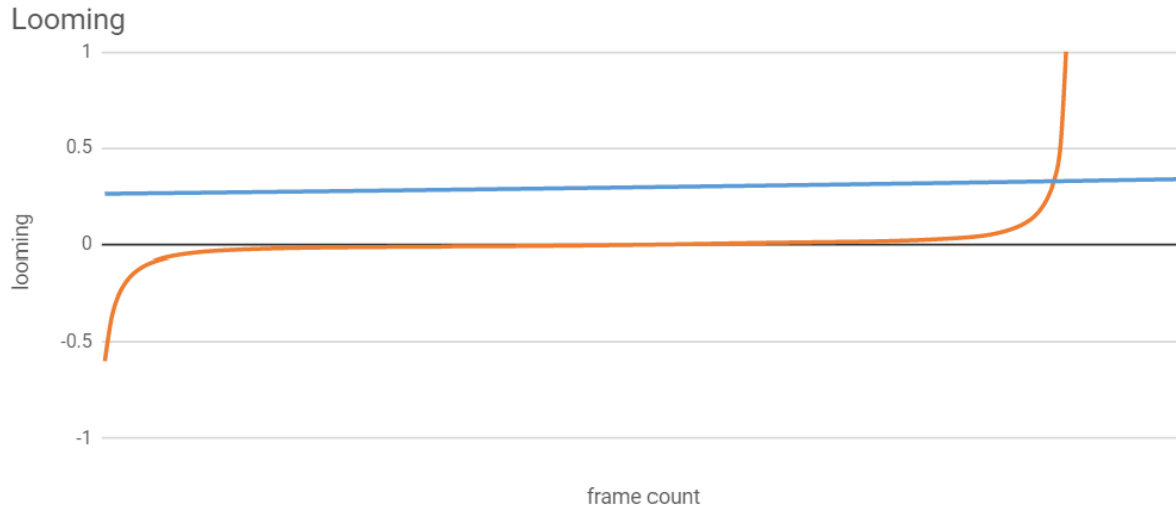


בהתחלה היא עולה למרות שהרכב מתקרב בגלל שאין עדיין כושר הפרדה בין האלמנטים בטקסטורה כשהרכב עוד רחוק, אבל החל משלב מסוים היא יורדת כצפוי.



להלן הגרף שמראה את שינוי ה looming כפונקציה של הזמן. בכתום מוצג הגרף של ערכי ה looming שחושבו בשיטה של צפיפויות טקסטורה, ובכחול בשיטה המדויקת של חישובי מרחקים.

שימו לב כיצד ה looming עולה אקספוננציאלית בגרף הכתום לעומת עלייה לינארית בגרף הכחול, וכאשר צפיפות הטקסטורה יורדת בגרף הקודם, ה- looming עולה בגרף הזה.



מסקנות

מההשוואה בין השיטות ניתן להסיק שבשיטה של צפיפות הטקסטורה ה looming גדל באופן שונה. הוא גדל באופן אקספוננציאלי בעוד שה looming שמחושב בשיטת המרחקים גדל באופן לינארי. בשיטת צפיפות הטקסטורה, כשהרכב עוד רחוק ה looming גדל מאוד לאט ורק כשהוא כמעט נפגש במכשול הוא עולה הרבה יותר מהר. זה יכול להוות בעיה אם רוצים לממש קבלת החלטות על בסיס ערכי ה looming האלו כי הרכב פונה כשהוא רואה שמלפניו יש מספיק אדום והאדום מופיע לאחר זמן רב ובמשך הפנייה של הרכב צריך לאפשר לו מרחב ביטחון כדי לסיים את הפנייה ללא התנגשות. הרי הוא מתקדם גם תוך כדי הפנייה, ואם הוא יתחיל לפנות מאוחר מידי הוא יתנגש במכשול.

המלצות

מאמר 1 (ראה מקורות ספרותיים) מתאר מספר שיטות שבעזרתן ניתן לחשב looming מתוך רצף של תמונות. למשל ניתן גם לחשב שינוי של שטחים של מכשולים כפונקציה של הזמן. אם נחשב looming לפי השיטות האלו ונאחד אותן, נוכל לקבל תוצאות טובות יותר, כי לאלגוריתם יהיה יותר מידע על הסביבה שלו מהתמונות.

עוד אפשרות היא להשתמש ברשת עצבית. בנספח 2 מצורף לינק לפיתוח של Deep Mind ובו תראו אנימציה של גוף קופץ מעל מכשולים, ולדעתנו כדאי ללמוד מהמאמר שלהם ולשלב ביחד עם השיטות המתוארות במאמר 1.

מקורות ספרותיים

1. The Visual Looming Navigation Cue: A Unified Approach
Daniel Raviv and Kunal Joarder
2. A New Method to Calculate Looming for Autonomous Obstacle Avoidance
Daniel Raviv and Kunal Joarder

נספחים

1. Hover-UI-Kit - User interfaces for immersive VR/AR experiences.
<https://github.com/aestheticinteractive/Hover-UI-Kit>
2. Deep Mind – Producing flexible behaviours in simulated environments.
<https://deepmind.com/blog/producing-flexible-behaviours-simulated-environments/>