שילוב עננים "מציאותיים" בסימולטורים

מבוא

עבודה זו בוצעה בהנחייתם המשותפת של ירון חונן מהטכניון (מעבדת GIP) וחברת רפאל.

בעבודה זו פיתחנו אלגוריתם שמטרתו לשלב עננים בסימולטורים (ספציפית – שילוב עננים בעלי נפח בסצנה של Unity).

המטרה הייתה לשלב עננים שייראו מציאותיים ככל הניתן, תוך התבססות על התזה:

Real-time rendering of volumetric clouds,UMEA University, Fredrik Haggstrom

בחרנו להתמקד בתזה הנ"ל עקב התוצאות המוצגות בה, אשר היו טובות יותר בהשוואה לגישות אחרות שניסינו קודם לכן (פירוט נוסף בפרקים הבאים).

מטרתו של מסמך זה הינה לשתף את הקורא באלגוריתם שפיתחנו, ובאתגרים השונים שנתקלנו בהם במהלך הפיתוח.

לרקע כללי יותר, מומלץ להשתמש במצגת שהכנו.

גישות שונות בדרך לפתרון

חשוב לציין כי עבודה זו התבצעה בשנת 2019, וכי קיים סיכוי שבעתיד אחת הגישות שזנחנו תתאים למטרות הדרושות לקורא.

Unity Particle System

בהתחלה ניסינו להשתמש ב- Unity Particle System אך לא מצאנו דרך לשלוט על התאורה של "כלל צורת הענן", דבר אשר היה הכרחי בעינינו כדי לדמות עננים מציאותיים. להלן סירטון הדגמת יכולות של גישה זו:

https://www.youtube.com/watch?v=BQcjsW8ldkw

Unity VFX Graph

בהמשך ניסינו להשתמש בכלי שמתבסס על הגישה הקודמת, שנקרא Unity Visual Effect Graph אשר אחד מיתרונותיו הבולטים הוא הקלות הרבה שבה הוא מאפשר למשתמש לשלוט על פרמטרים שונים, אך גם כיוון זה נשלל ברגע שהבנו שלא ניתן לשלב shader שייתן לנו לעשות מניפולציות על הצבע של כלל צורת הענן.

להלן הדגמת יכולות של גישה זו:

https://www.youtube.com/watch?v=R6D1b7zZHHA&t=290s

Ray marching (as part of the thesis we followed)

בשלב יחסית מוקדם בקריאת התזה, הבנו כי זה הכיוון הכי מבטיח, כיוון שהוצגו בו תוצאות טובות יחסית לכיוונים הקודמים שנבדקו. כיוון זה הציג גישה שמשלבת Ray-marching , טקסטורות, ומניפולציות מתמטיות בדרך לשילוב עננים מציאותיים ככל הניתן בסצנות. עם זאת, גם לגישה זו קיימים חסרונות.

נציין את העיקריים שבהם:

1. הפתרון לא נבנה ב-Unity.
2. אין פירוט של הערכים המושמים בכלל הפרמטרים, בעוד שקיים מספר רב מאד של פרמטרים, ולכל פרמטר קיים תחום רחב של ערכים.
3. שלבי אופטימיזציה מסוימים מצוינים רק ככותרת ואינם מוסברים.
4. קיים שימוש במונחים "ידועים" אשר מומשו בדרך שונה במאמר. לדוגמא המונח Ray-marching ממומש במאמר זה באופן מסוים, שאינו אופן המימוש שהיה מוכר לנו.

בסופו של דבר החלטנו להתקדם עם גישה זו על אף חסרונותיה, כיוון שבחרנו להסתכל על החסרונות בתור אתגרים שמשיקים לנושאים שעניין אותנו להתנסות בהם: Unity, Shaders, Ray-marching, Textures, Clouds.

רקע תאורטי

בפרק זה נסביר מעט על הרקע של התזה שמימשנו:

Real-time rendering of volumetric clouds,UMEA University, Fredrik Haggstrom

עננים

רוב המודלים ליצירת עננים לא מתבססים על פיזיקה שמדמה את התנהגות העננים בעולם האמיתי. עם זאת, קיימת חשיבות עליונה להבין כיצד מתנהגים העננים בעולם האמיתי כדי ליצור מודל דומה ממוחשב, כיוון שמאפייני העננים בעולם האמיתי מהווים את הבסיס לכל פתרון שניצור במחשב.

עננים מורכבים מטיפות מיים או קרח קריסטלי בגודל כלכך קטן שכל מטר מרובע מכיל כ-100 מליון טיפות. הגובה והטמפרטורה שישררו בו יהיו הגורם העיקרי שיכריע האם הטיפות יהיו במצב צבירה מימי או קרחי.

האוויר תמיד מכיל אדי מיים, אשר עולים למעלה בגובה, ואז מתקררים למצב צבירה נוזלי, אך כדי שיצטברו לכדי "מבנה" שדומה לענן, דרושים חלקיקים נוספים כגון מלח או אבק.

- לרקע מורחב שכולל הסבר על סוגי העננים השונים וצורות התאורה המשתלבות עם העננים יש לפנות לתזה.

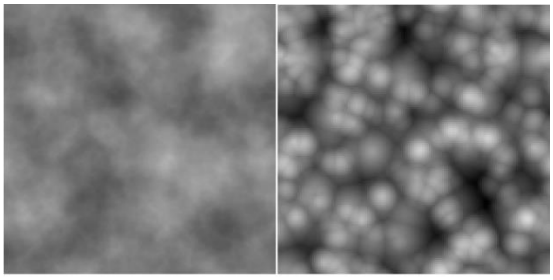
יצירת מודל ממוחשב של עננים

אפשר "לרנדר" עננים ע"י שיטות שונות כגון תצלום עננים אמיתיים והקרנתם בסצנה, אך הבעיה עולה כאשר רוצים ליצור עננים שבהם קיימים אזורים עם צפיפות שונה (באותו הענן).

במצב כזה, נרצה שהענן יוגדר ע"י נקודות ספציפיות ולא ע"י פוליגון. ברגע שהענן מתחיל להיות מוגדר ע"י נקודות במקום פוליגון, עולה הצורך "לרנדר" אותו באופן אחר, כאשר השיטה שהוצעה ב-2014 ע"י מונוז נקראת "ray-marching". כיום זוהי שיטה פופולרית מאד לדגימת צפיפות של עצמים מרונדרים, כאשר מהותה כוללת עקיבה אחרי הקרניים שיוצאות מהמצלמה לתוך הסצנה.

אחת השיטות הפופולריות היא יצירת עננים בעזרת טקסטורות בדו ובתלת מימד, אשר יוצרות ביחד צורה של עננים.

כדוגמא- אם ניקח פונקציית INVERTED WORLEY (הפונקציה ההופכית הדרושה ליצירת עיגולים לבנים, כיוון שהפונקציה הרגילה יוצרת עננים שחורים). היא תיצור לנו את הצורה הכללית הדומה לענן- כדורים כאלה אחד בתוך השני, ואם נוסיף את פונקציית הרעש Perlin אז נקבל צורת "ערפל" אשר תתלבש על הטקסטורה הקודמת, כל שביחד תיווצר טקסטורה דמוית ענן.



(a) Perlin noise. (b) Inverted Worley noise.

Figure 5: Examples of Perlin and Worley noise.

תיאור המערכת

בחרנו להציג את המערכת שלנו בגישת Top Down כיוון שמניסיוננו זוהי הדרך הכי אינטואיטיבית להבין את מה שעשינו.

מי שמעונין לקרוא על המערכת באופן שמתחיל מהפונקציות הפנימיות מוזמן לפנות למצגת שהכנו.

נדבך נוסף שלא נרד לעומקו בדו"ח זה הינו מבנה הפונקציות המתמטיות. מי שמעונין להבין את הפונקציות המתמטיות מוזמן לפנות לתזה ולהשוותה לפונקציות בקוד שלנו.

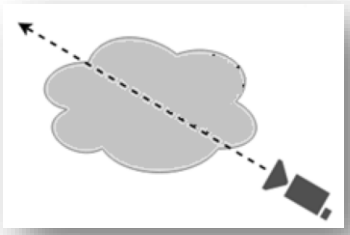
השלבים המוצגים בהמשך הינם השלבים העיקריים בראייתנו. לכלל השלבים נא לפנות למצגת ולתזה.

רקע הכרחי להבנת המערכת

- חשוב להבין כי העננים מרונדרים למסך בעזרת סקריפט של שיידר, אשר מוצמד לעצם שמוגדר כ"מצלמה", כלומר **אין** עצם תלת מימדי פיזי של "עננים".
- דרוש להבדיל בין שלבי החישוב שנוגעים לעננים מבחינת צורה וצבע, לבין שלבי הצגת העננים אשר משלבים את העננים בתוך הסצנה.
- כיוון שכל פיקסל במסך מורכב מ-4 ערוצים- RGBA, אז צורת הענן תיקבע עפ"י ערוץ ה-A, וצבע הענן ייקבע עפ"י ערוצי ה-RGB.
- החישוב "כיצד ייראו העננים ומה יהיה צבעם" מתבצע ללא תלות בעצמים הגרפיים בסצנה, פרט לעצם שמוגדר כ-"שמש", ולעצם שמוגדר כ"מצלמה".
- לאחר שהחישוב לגבי צורת וצבע העננים מסתיים, כלומר לאחר שסיימנו לחשב את ערוצי ה-RGBA עבור כל הפיקסלים במצלמה, והבנו באיזה מהם יופיע ענן-
מחשבים "מה מסתיר את מה" בסצנה, ורק לאחר מכן ניתן להציג את העננים.

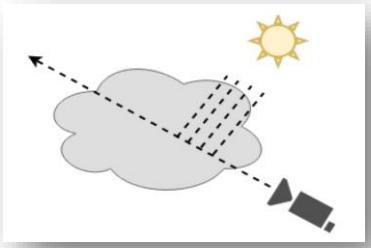
- חישוב ערך ערוץ ה-A של הענן (שקול לצפיפות הענן, ולמידת שקיפותו).
  a. מסתמך על טקסטורות שונות, נוסחאות מתמטיות, ומימוש וריאציה של Ray-marching מכיוון המצלמה לכיוון שאליו היא מסתכלת.

RGBA



- חישוב ערך ערוצי ה-RGB של הענן (שקול לצבע הענן).
  a. מסתמך על נוסחאות מתמטיות, ומימוש וריאציה של Ray-marching אשר מתבצעת משני מקורות
     - כנ"ל בצעד 1
     - מכיוון הפיקסל הספציפי בענן שרוצים לחשב את צבעו, לכיוון העצם שהגדרנו כ"שמש".

RGBA



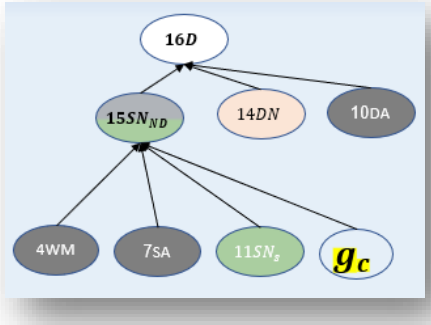
להלן פירוט שלבי האלגוריתם ממצגת הסיכום:

- נא לשים לב כי המספור של השלבים לא תואם בהכרח למספור בדו"ח זה. הדבר נובע מהאופן השונה שבו רצינו להדגיש שלבים שונים באלגוריתם. בדו"ח זה אנו מתעמקים יותר בשלבים הממוספרים בתמונה הנ"ל כ: 1-3.

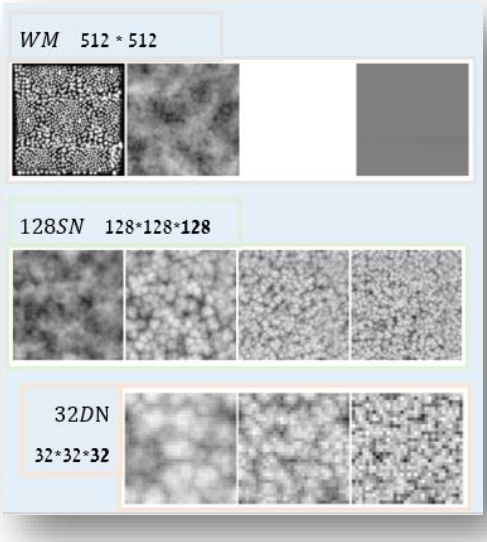
הערה- פירוט על כלל הנוסחאות המתמטיות נמצא בפרק "הנספחים" בסוף הדו"ח. מספרי הנוסחאות באיורים למטה תואמים למספר הנוסחה שתופיע בפרק הנספחים.

1. חישוב ערך ערוץ ה-A של הענן (שקול לצפיפות הענן, ולמידת שקיפותו).

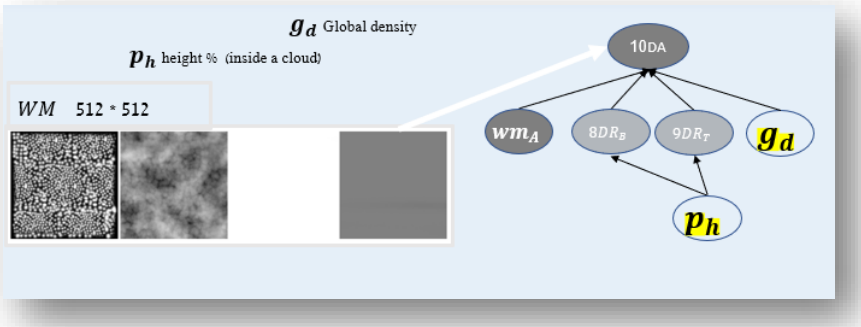
- הטווח של ערך זה עבור כל פיקסל יהיה בין אפס לאחד, כאשר אפס מעיד על כך שאין ענן כלל בפיקסל זה, ואחד מעיד על כך שצפיפותו של הענן מלאה בפיקסל זה.
- כל פיקסל מקבל ערך "צפיפות" לפי נוסחה 16, אשר מתבססת על גרף הנוסחאות הבא:

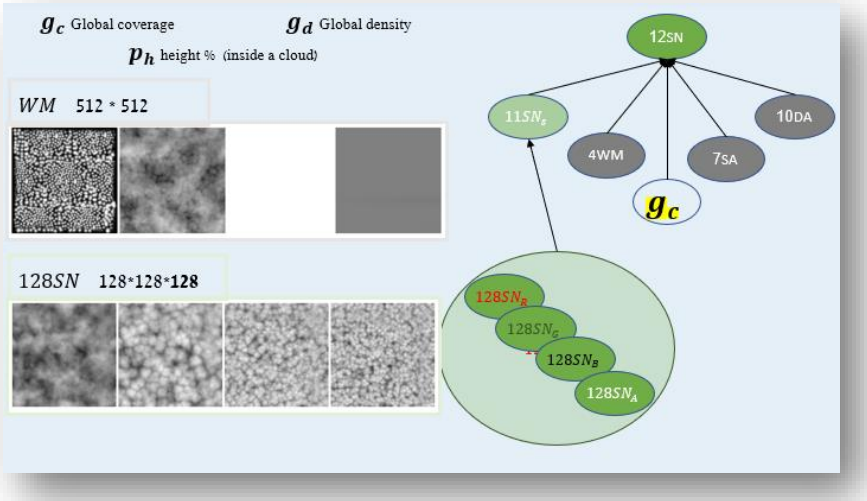


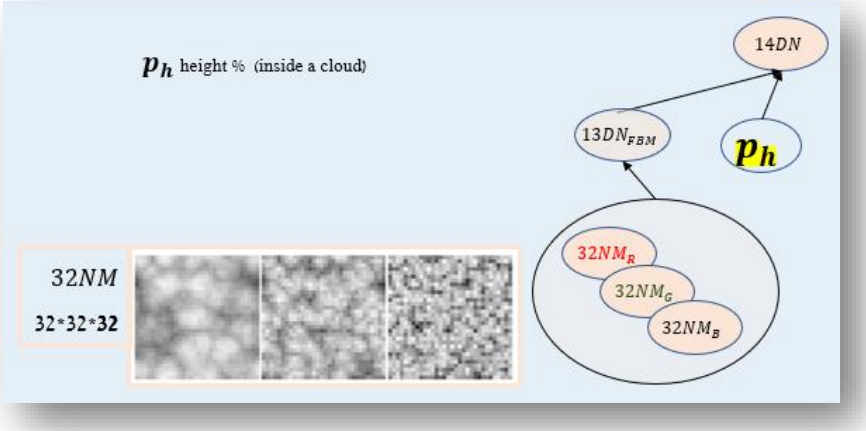
- כל צבע מתייחס לאחת הטקסטורות הבאות:



- כל פיקסל בכל אחת מהטקסטורות מספק ערך בטווח [0-1], אשר משולב באחת הנוסחאות. (למידע נוסף על הטקסטורות פנה למצגת או לתזה).
- להלן מספר דוגמאות אשר מראות כיצד הטקסטורות משתלבות בנוסחאות המתמטיות:

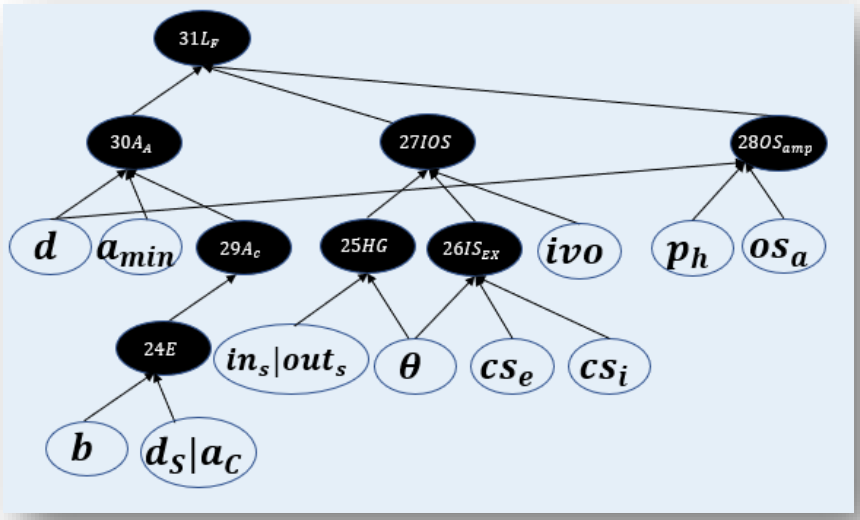

g_d Global density
p_h height % (inside a cloud)
WM 512 * 512
10DA
wm_A
$8DR_B$
$9DR_T$
g_d
p_h


g_c Global coverage
g_d Global density
p_h height % (inside a cloud)
WM 512 * 512
128SN 128*128*128
12SN
$11SN_E$
10DA
4WM
7SA
g_c
$128SN_R$
$128SN_G$
$128SN_B$
$128SN_A$


p_h height % (inside a cloud)
14DN
$13DN_{FBM}$
p_h
32NM
32*32*32
$32NM_R$
$32NM_G$
$32NM_B$

2. חישוב ערך ערוצי ה-RGB של הענן (שקול לצבע הענן).

- באופן דומה, גם כאן אנו רוצים להגיע לערך מתמטי שייצג צבע של ענן מציאותי, וכך יבליט מאפיינים נוספים בו. השגת מטרה זו מתבצעת ע"י תוצאת פונקציה 31. (הגרף הבא יראה על איזה פונקציות נוספות היא מסתמכת)
- <u>אנחנו מחשבים ערך שמטרתו לייצג עד כמה כהה פיקסל מסויים בענן. לאחר מכן הערך הזה מוכפל במשתנה גלובאלי המייצג את צבע האור של השמש, כך שבסופו של דבר הצבע של כל פיקסל בענן מוגדר כמכפלה של שני אלה.</u>
  - אם הערך שחושב הוא 0 או ערך קרוב לכך – צבע הפיקסל יהיה שחור.
  - אם הערך שחושב הוא 1 או ערך קרוב לכך- צבע הפיקסל יהיה לבן.
  - בכל וריאציה אחרת נקבל גוון מסוים של צבע השמש.
- יש לציין כי רוב קטע הקוד שאמון על ערך ערוצי ה-RGB נמצא בנספח שבתזה.



- <u>לצורך הבנת מהות המשתנים נא לפנות לפרק הנספחים.</u>

אתגרים עיקריים

המוטיב החוזר ברוב האתגרים שנציין הוא כמות הפרמטרים הרבה, והטווח הגדול של כל פרמטר. במאמר לא ירדו לרזולוציות אלה, ולכן ברוב המקרים החלטנו על השמת ערכים לפרמטרים עפ״י ניסוי וטעיה. לא מין הנמנע כי ע״י בדיקות נוספות שיכללו בידוד ובדיקת כל פרמטר בנפרד- ניתן ליצור עננים מציאותיים יותר.

אתגר כללי כמובן שנתקלנו בו הוא ניסיון מועט ב-Unity ובשיידרים ו-Ray-marching בפרט.

השתמשנו בסדרת סרטונים ביוטיוב למימוש תשתית בסיסית של Ray-marching, אך לאחר זאת מימשנו את הכל באופן עצמאי לחלוטין- מה שדרש הרבה ניסוי וטעיה.

צורת העננים וצבעם

1. <u>השוואה לתוצאות המאמר-</u> לאורך המאמר, הכותב מציג תמונות שנראות טוב יותר ככל שהוא מוסיף עוד פונקציות ומרחיב את המערכת, אך התמונות הללו כוללות כבר חלק ממנגנון ה״תאורה״ שאחראי גם על הצבע (ה-RGB) של העננים, לכן במידה וענן מסוים נראה עם פחות מאפיינים ממה שמוצג במאמר, לא מין הנמנע כי הדבר קשור לתאורה.
2. <u>טקסטורות-</u> טקסטורות שונות משפיעות לבלי היכר על הצורה המתקבלת של העננים. המפה הראשית 512WM צוירה ע״י אדם בתוכנה גרפית (כגון Photoshop), ושאר המפות- 128SN, 32DN נבנו ע״י רעשי Perlin, Worley והשילוב של שניהם.
כאשר ניסינו להשתמש במפות ה- 512WM קיבלנו תוצאות שונות מהתוצאות שהוצגו במאמר, אך לא ניתן לקבוע בוודאות מה הסיבה לכך כיוון שמפות ה- 128SN & 32DN כוללות מימד שלישי שלא ניתן לנו (לדוגמא המפה האחראית על קילוף צורת הענן בנויה מ-128 שכבות). כלומר איננו יודעים מה הפרמטרים המדויקים ששימשו את כותב המאמר בעת שבנה את המימד השלישי, ומניסויים שביצענו, כל שינוי של פרמטר בבניית הרעש שינה באופן משמעותי את צורתם של העננים.
3. <u>Ray-marching-</u>

- כמות הצעדים שמבצעים, והאופן שבו מבצעים אותם משפיעה גם כן באופן רב על צורת העננים. האלגוריתם מציב ערך מתמטי של צפיפות עבור כל פיקסל בשכבת העננים. כאשר המצלמה מסתכלת לכיוון העננים, היא תתחיל לדגום ערכים בוקטור התקדמות מסוים, אך כיוון שבמאמר לא מצוין מהי כמות הצעדים המקסימלית שמבוצעת מכל פיקסל במצלמה, היינו צריכים ע״י ניסוי וטעיה להציב ערך מסוים בעצמנו (לשם פיתוח בחרנו שיהיה 64). במצב אידאלי, היינו בוחרים ערך גדול ככל שאפשר כי לא נרצה להגיע למצב בו אנחנו עוצרים את ה-Ray Marching מוקדם מידי, לפני שנדגם כל הענן. בפועל, גילינו שערכים גדולים אכן גורמים לשיפור בנראות, אך פוגעים דראסטית בכמות הזמן שלוקח ל-Unity לקמפל את ה-shader ולחשב שינויים בקוד. לכן, מסיבות פרקטיות ולשם פיתוח מהיר יותר על המחשבים האישיים שלנו, בחרנו בערך נמוך יחסית.
- אורכם של הצעדים גם כן השפיע רבות על נראות העננים. במאמר לא מצוין מה גודלם של הצעדים שנלקחו, ולכן גם פה היינו צריכים ע״י ניסוי וטעיה להציב ערכים שגרמו לעננים להיראות מציאותיים יותר.

- מימוש האופטימיזציות שנגעו לנושא זה בהחלט שיפרו את זמן הריצה. ללא אופטימיזציות כלל המחשבים שלנו התקשו לרנדר את הסצנה, אך לאחר ביצוע האופטימיזציות הבאות Unity רצה באופן סביר:
  - הפסקת עקיבת הקרניים כשמגיעים לצפיפות מלאה
  - ביצוע עקיבת קרניים רק כאשר נמצאים בגבהים של שכבת העננים
  - ביצוע צעדים אשר נהיים גדולים יותר כאשר מתרחקים יותר מהמצלמה
  - שינוי גודל הצעד בהתאם למיקום: בתוך ענן צעדים קטנים יותר, מחוץ לענן צעדים גדולים יותר

4. <u>היחס שבין הגורמים הבאים: מיקום המצלמה, גובה שכבת העננים, כמות הצעדים המירבית של עקיבת הקרניים</u> – אנו קיבענו את גובה שכבת העננים ל- [1000-400] אך יש לציין כי הגובה המינימלי והמקסימלי משפיעים רבות על נראותם. ככל שהעננים רחוקים יותר מהמצלמה נדרשים יותר צעדי עקיבת קרניים כדי לדגום צפיפות של עננים, הדבר ישפיע באופן ישיר על כמות הזמן הנדרשת לחשבם, בשילוב לכך שקבענו "סף עליון" לכמות הצעדים שמבצעים- יכול להיות שכלל לא נראה עננים פשוט מהסיבה שלא מבוצעים מספיק צעדים בתוך שכבת העננים.

פרמטרים שונים והשפעתם על העננים

<u>כללי-</u>

3. יש לשים לב כי קיים קשר בין הסקריפט שמוצמד למצלמה, לבין הסקריפט של השיידר עצמו. כלומר אחד האתגרים באופן כללי היה להבין כיצד משתמשים בסקריפט של שיידר, או ליתר דיוק כיצד להעביר אליו משתנים חשובים. RaymarchCamera הוא הסקריפט שמוצמד למצלמה, אשר מחזיק בתוכו רפרנס לשיידר עצמו שנקרא RaymarchShader.
4. מעבר למשתנים הבאים, יש לשים לב כי קיימים משתנים פנימיים נוספים בתוך הסקריפט של השיידר שגם כן משפיעים על העננים (כגון המשתנה שקובע סף עליון לכמות צעדי עקיבת הקרניים), ובנוסף קיימים משתנים בעצם שמוגדר כ"שמש" שגם כן ישפיעו על נראות העננים (כגון המשתנה Intensity).

המשתנים הבאים חשופים ב-EDITOR ומוצמדים למצלמה:

- <u>Shader-</u> סקריפט אשר אחראי על השיידר, בתוכו נמצאים כלל החישובים שביצענו.
- <u>Cloud Layer Height-</u> גובה מינימלי ומקסימלי של העננים שיוצרו.
- <u>Global Cloud Coverage-</u> עד כמה שכבת העננים תהיה מכוסה בעננים. מבוססת בגדול על טקסטורת ה-512WM (בנויה מ-RGBA) כאשר עד ערך של 0.5 דוגמים מערוץ ה-R ומעל לערך של 0.5 דוגמים גם מערוץ ה-G.
- <u>Global Cloud Density-</u> ערך זה מגדיל את הסבירות שהעננים יהיו בעלי צפיפות גבוהה יותר. צפיפות גבוהה יותר משפיעה גם על צבע הענן וגם על הצורה שלו.

- Weather Map Scale- משפיע על האופן שבו תידגם טקסטורת ה-512WM. תלוי בטקסטורה, לפעמים ערכים נמוכים שקרובים ל-0 יתנו נראות טובה של עננים ולפעמים ערכים כגון 6000 ומעלה יתנו נראות רצויה.
- Offset X/Y- שינוי הערכים יזיז את שכבת העננים. (בעזרת שינוי ה"אופסט" ניתן לשנות את צורת העננים ולגרום לאשליה שהם זזים. ספציפית במימוש שלנו שינוי האופסט מזיז רק את טקסטורת ה-512WM אך ניתן לשנות זאת בעתיד כך שיזיז את כלל הטקסטורות יחדיו, למידע נוסף פנה למצגת או לתזה).
- Shape Noise Scale – משפיע על אופן הדגימה מהטקסטורה **התלת מימדית** 128SN (בגודל 128*128*128) ובכך בעצם משפיע על צורת העננים באופן ניכר. הערך לדוגמא אשר נתן עננים יחסית מציאותיים היה 3000 אך חשוב לזכר ששילוב ערכים של פרמטרים שונים משפיע גם כן על הנראות ולכן יש להיות פתוחים לכך שערך שונה לחלוטין ייתן תוצאה טובה יותר בעתיד.
- Weather Map – כאן יש להזין את מפת מזג האוויר אשר תשפיע על המיקום שבו ייווצרו עננים. זוהי תמונה בפורמט TIF בגודל 512*512 פיקסלים בעלת ארבעה ערוצי קלט - RGBA. לפרטים נוספים יש לפנות למצגת או לתזה.
- Blue Noise – תמונה מסוג PNG בגודל 512*512 בעלת 3 ערוצים RGB ומטרתה לרכך את המעבר שבין קצוות הענן לבין הרקע שסביבו, ולמנוע Banding. אין צורך לשנות את התמונה הזו בשימוש יום-יומי. לפרטים נוספים יש לפנות לתזה.
- Detail Noise Scale – משפיע על האופן שבו תידגם הטקסטורה **התלת מימדית** 32DN. ישנה את הנראות של הענן בצורה עדינה אך נראית לעיין. יכול להוסיף לנראות מציאותית יותר גם בערכים של 3 אלף או 30 אלף. יש לשחק עם פרמטר זה ולראות כיצד הוא משתלב בפרמטרים האחרים.
- Lighting Cloud In/Out Scatter – שני הפרמטרים האלה משפיעים רבות על האופן שבו העננים שנמצאים בקו ראיה ישיר מהמצלמה לשמש נראים. הם עוזרים להבליט את מיקום השמש, כיוון שבעזרתם עננים שסמוכים יותר לשמש יהיו בגוון שונה מעננים שאינם ליד השמש. בעזרת הערך 0.51 ב-In והערך 0 ב-Out קיבלנו תוצאה יחסית יפה.
- Lighting beer – משתנה אמנותי שבא לדמות חלק מהנוסחה של מעבר "אור" דרך תווך. במקרה שלנו דרך העננים. לפי התזה ערכו של משתנה זה צריך להיות 12, אך הקורא מוזמן לבדק ערכים אחרים ולראות כיצד זה משפיע על צבע העננים.
- Lighting In VS Lighting Out Scatter – אינטרפולציה ליניארית בין שני הערכים Lighting Cloud In/Out Scatter. בשילוב עם הערכים שציינתי קודם, כששמנו פה ערך ששווה ל-0.5 קיבלנו איזון יפה בין שני סוגי הפיזור של האור.
- Lighting Silver Lining Intensity – עצמת הזוהר שקשורה לקרני האור שיחצו דרך העננים מהשמש. אצלנו הערך היה 2.5, למידע נוסף פנה לתזה.
- Lighting Silver Lining Exponent – פרמטר נוסף עד כמה הזוהר יהיה סביב השמש במרחב הקרוב אליה או הרחוק ממנה. אצלנו נקבע על 0.2.
- Lighting Ambient out Scatter – הוספת פרטים לאזורים מאד מוארים, טווח [0-1]
- Lighting Attenuation Clamp Val – עד כמה העננים יכולים להיות שחורים. טווח [0-1]
- Lighting Minimum Attenuation – תוסיף פרטים לעננים כהים. טווח [0-1]

התרומה שלנו

1. <u>בניית מערכת שלמה מקצה לקצה המבוססת על התזה-</u> בניית המערכת מקצה לקצה תקל מאד על כל הבאים אחרינו. זה היה אתגר לא פשוט להבין את כוונתו של כותב התזה, ואז להפוך את ההבנה הזאת לקוד שעובד. מעתה כל מי שירצה להתקדם בכיוון שהתחלנו בו, יוכל להשוות בין המאמר לפרויקט שלנו, ובכך להבין טוב יותר ומהר יותר כיצד מה שכתוב במאמר בא לידי ביטוי בקוד. זהו יתרון עצום בבואך להבין נושא מסוים ואז לשפרו.
   - דוגמא לשניים מהדברים הרבים שהתמודדנו עימם ללא פרטים מהתזה הוא המאפיינים של שכבות הרעש התלת מימדיות, והשילוב של זוויות קרני השמש והשפעתם על נראות העננים.

2. <u>נראות עננים טובה באופן מידי-</u> גם לאחר מימוש המערכת, עדיין נותרה המון עבודה כיוון שלא צוין בתזה אילו ערכים לשים לכלל הפרמטרים. מי שימשיך להתעסק עם הפרויקט אחרינו- יקבל מערכת בנויה עם סט ערכים "ברירת מחדל" שכבר מראה עננים ברגע פתיחת הפרויקט. (כאשר הפרמטרים לא מכוונים כראוי לא רואים עננים!).

3. <u>שינוי פרמטרים באופן חופשי-</u> כהמשך ישיר של הנקודה הקודמת, כעת ניתן באופן מידי לשנות את ערכי הפרמטרים ולראות את השפעתם, באמצעות ממשק נוח למשתמש, בלי צורך לשנות את הקוד. כל המשתנים החשובים שמשפיעים על נראות העננים חשופים למשתמש באמצעות ה-inspector, וכל המשתנים הפנימיים שקשורים לריצת האלגוריתם מוגדרים כקבועים בקוד וניתן לשנותם בקלות. אין צורך לנחש מה ההשפעה של משתנה כלשהו מהתזה כאשר אפשר ישירות לשנות את המימוש הקיים ולראות באופן עצמאי.

4. <u>Import as Package -</u> התרומה האחרונה שנציין היא שכעת ניתן להוסיף את העבודה שביצענו לכל סצנה בפשטות, כיוון שהפכנו את כל מה שנדרש עבור שכבת העננים ל"Package" שניתן פשוט לייבא.

   - צעדים לשימוש בשכבת העננים:

1. Assets -> Import Package -> Custom Package -> Choose “KT_Clouds_Package -> Open
2. Import All
3. Drag the “CloudRenderer” (has a cloud icon on it) to the Main Camera
4. Adjust the parameters according to your scene
   - Note: The Light in your scene influences as well on the Clouds

משימת עתידיות

1. המשימה הראשונה שאנו ממליצים עליה היא העמקה בטקסטורות. הטקסטורות השונות הן אלה שאחראיות בצורה ניכרת על צורת העננים, ואנו בטוחים שע״י העמקה באופן בנייתן ניתן להגיע לעננים שייראו מציאותיים יותר. לדוגמא: כל טקסטורה בנויה מרעש מסוים, לכל רעש כזה יש מספר פרמטרים. ע״י הבנה עמוקה יותר בהשפעת הפרמטרים וביצוע ניסוי וטעיה יהיה ניתן למצוא את הערכים הכי מתאימים עבור כלל סוגי העננים.
2. המשימה השנייה שאנו ממליצים עליה היא מימוש אלגוריתם ה- Re-Projection* (לינק לפורום שיכול להוות נקודת פתיחה טובה בנושא). אלגוריתם זה מוריד משמעותית את הזמן שלוקח לרנדר את הסצנה, ולא ההספקנו להתקדם בנושא זה כלל.
3. המשימה השלישית שאנו ממליצים עליה היא שיפור אופן יצירת ותזוזת העננים כך שינועו באופן מציאותי יותר.

---

*Schneider, A. (2016). GPU Pro 7, Real-time Volumetric Cloudscapes, chapter 4, pages 97–127. CRC press.

סיכום

המימוש הנוכחי מהווה נקודת פתיחה טובה למי שירצו לשפרו אחרינו, שכן הרבה יותר קל לראות מערכת מסוימת ולהסיק ממנה מסקנות על חלקיה השונים מאשר להתחיל לבנות אותה מאפס כפי שאנו בנינו אותה.

נציין כי כחלק מהניסיונות לשפר את תוצאות המימוש שלנו, פנינו לכותב התזה אך לא הצלחנו להשיגו. לכן חשוב לנו להדגיש כי נשמח לתרום לכל מי שימשיך את עבודתנו. בתור השלמה לדו"ח זה, ניתן לפנות גם למצגת שהפצנו, לסרטון שהכנו, וגם כמובן לתזה שהתבססנו עליה.

בנוסף ברצוננו להודות לירון חונן ממעבדת GIP בטכניון, ולחונכים שלנו מרפאל, אשר גם עזרו למקד אותנו ע"י הגדרה מעולה של המשימה, וגם תמכו בנו כאשר הבענו רצון לממש את התזה הנ"ל במסגרת זמנים קצרה יחסית.

מקווים שעבודתנו תועיל לקידום פיתוחם של אמצעים שיעזרו להגן על אזרחי מדינתנו.

- אריק ולב.

- בנספח זה מפורטות כלל הפונקציות שציינתי בדו"ח. לרקע נוסף ניתן לפנות לתזה.

A.1 Global parameters

$g_c \in [0,1]$, global coverage, controls the basic probability for clouds to appear.
$g_c \in [0,\infty]$, global density, determents the clouds opacity.
$p_h \in [0,1]$ is the height percentage of where (in altitude) in the cloud the current sample point is.

$w_h \in [0,1]$, the cloud maximum height value from the weather-map's blue color channel.
$w_d \in [0,1]$, the cloud density value from the weather-map's alpha channel.

Function 1 converts/remaps a value from one range to another, where $v \in \mathbb{R}$ is the value to be remapped, $l_o \in \mathbb{R}$ is the left endpoint of original range, $h_o \in \mathbb{R}$ is the right endpoint of original range, $l_n \in \mathbb{R}$ is the left endpoint of new range, and $h_n \in \mathbb{R}$ is the right endpoint of new range.

$$R(v,l_o,h_o,l_n,h_n) = l_n + \frac{(v-l_o)\times(h_n-l_n)}{h_o-l_o} \tag{1}$$

Function 2 clamps the value $v \in \mathbb{R}$ to be in the range 0 to 1.

$$SAT(v) = \begin{cases} 0 & \text{if } v < 0 \\ 1 & \text{if } v > 1 \\ v & \text{otherwise} \end{cases} \tag{2}$$

Function 3 linearly interpolates between $v_0 \in \mathbb{R}$ and $v_1 \in \mathbb{R}$ by an amount of $i_{val} \in \mathbb{R}$. With $i_{val} = 0$, the function will return v_0, with $i_{val} = 1$, the function will return v_1, values in-between are interpolated, and values outside the range $[0,1]$ are extrapolated.

$$L_i(v_0,v_1,i_{val}) = (1-i_{val})\times v_0 + i_{val}\times v_1 \tag{3}$$

$$WM_c = \max(w_{c0},\ SAT(g_c - 0.5) \times w_{c1} \times 2) \tag{4}$$

$$SR_b = SAT(R(p_h,\ 0,\ 0.07,\ 0,\ 1)) \tag{5}$$

$$SR_t = SAT(R(p_h,\ w_h \times 0.2,\ w_h,\ 1,\ 0)) \tag{6}$$

$$SA = SR_b \times SR_t \tag{7}$$

$$DR_b = p_h \times SAT(R(p_h,\ 0,\ 0.15,\ 0,\ 1)) \tag{8}$$

$$DR_t = SAT(R(p_h,\ 0.9,\ 1.0,\ 1,\ 0))) \tag{9}$$

$$DA = g_d \times DR_b \times DR_t \times w_d \times 2 \tag{10}$$

$$SN_{sample} = R(sn_r,\ (sn_g \times 0.625 + sn_b \times 0.25 + sn_a \times 0.125) - 1,\ 1,\ 0,\ 1) \tag{11}$$

$$SN = SAT(R(SN_{sample} \times SA,\ 1 - g_c \times WM_c,\ 1,\ 0,\ 1)) \times DA \tag{12}$$

$$DN_{fbm} = dn_r \times 0.625 + dn_g \times 0.25 + dn_b \times 0.125 \tag{13}$$

$$DN_{mod} = 0.35 \times e^{-g_c \times 0.75} \times L_i(DN_{fbm},\ 1 - DN_{fbm},\ SAT(p_h \times 5)) \tag{14}$$

$$SN_{nd} = SAT(R(SN_{sample} \times SA,\ 1 - g_c \times WM_c,\ 1,\ 0,\ 1)) \tag{15}$$

$$d = SAT(R(SN_{nd},\ DN_{mod},\ 1,\ 0,\ 1))) \times DA \tag{16}$$

$$E(b, d_s) = e^{-b \times d_s} \tag{24}$$

$$HG(\theta, g) = \frac{1}{4\pi} \frac{1 - g^2}{[1 + g^2 - 2g\ cos(\theta)]^{3/2}} \tag{25}$$

$$IS_{extra}(\theta) = cs_i \times SAT(\theta)^{cs_e} \tag{26}$$

$$IOS(\theta) = L_i(max(HG(\theta, in_s),\ IS_{extra}(\theta)),\ HG(\theta, -out_s),\ ivo) \tag{27}$$

$$OS_{ambient} = 1 - SAT(os_a \times d^{R(p_h, 0.3, 0.9, 0.5, 1.0)}) \times (SAT(R(p_h, 0, 0.3, 0.8, 1.0)^{0.8}))) \tag{28}$$

$$A_{clamp} = max(E(b, d_s), E(b, a_c))) \tag{29}$$

$$A_{alter} = max(d \times a_{min},\ A_{clamp}) \tag{30}$$

3.3.2 Lighting summary

There are several lighting values that can be changed to alter the clouds appearance:

- **Beer term** $\in [0,\infty]$, (b in function 24).
  Determines the amount of light attenuation.
- **In-scatter** $\in [0,1]$, (in_s in function 27).
  Determines the amount of light that is scattered trough the clouds.
- **Out-scatter** $\in [0,1]$, (out_s in function 27).
  Determines the amount of light that is scattered back the direction the light entered the clouds.
- **In- versus out-scatter** $\in [0,1]$, (ivo in function 27).
  Linear interpolation between **In-scatter** and **Out-scatter**.
- **Extra silver-lining intensity** $\in [0,1]$, (cs_i in function 26).
  Determines how extra bright the surrounding area around the sun should be.
- **Extra silver-lining exponent** $\in [0,1]$, (cs_e in function 26).
  Determines how centralized around the sun the extra brightness should be.
- **Ambient out-scatter** $\in [0,1]$, (os_a in function 28).
  Adds detail to very bright areas.
- **Attenuation clamp value** $\in [0,1]$, (a_c in function 29).
  Determines the how dark the clouds can be (clamps to a maximum attenuation).
- **Minimum attenuation ambient** $\in [0,1]$, (a_{min} in function 30).
  Adds detail to dark areas.

The final lighting function 31 is a combination of all three lighting functions (attenuation, in-/out-scatter, and out-scatter ambient).

$$L_{final}(\theta) = A_{alter} \times IOS(\theta) \times OS_{ambient} \tag{31}$$

מימוש ה-Ray marching בתזה

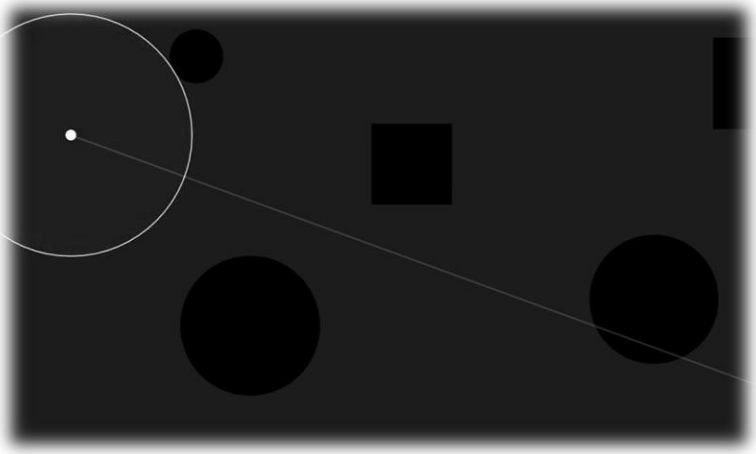
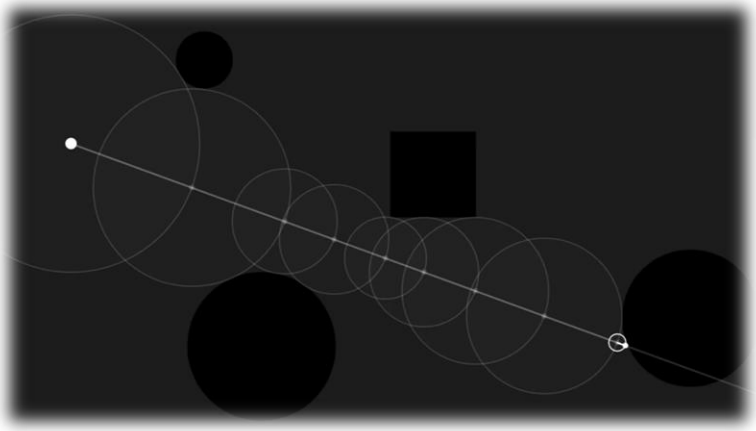
אופן המימוש מוסבר בפרק "תיאור המערכת", לכן פה ניתן רק תיאור תמציתי.

המימוש בתזה זו מתבצע באופן הבא:

עקיבת הקרניים מתבצעת מכל פיקסל במסך המצלמה שלנו באותו וקטור לכיוון העננים, כאשר המטרה היא להבין האם יש ענן בפיקסל זה ובאיזה שקיפות וצבע לרנדר אותו.

לצורך כך אנחנו מחליטים על גודל מסוים של צעד, אשר משתנה בהתאם לסיטואציות מסוימות, ובכל צעד אנו צוברים את דגימת הצפיפות העכשווית.

האלגוריתם משנה את גודל הצעד שלו בהתאם למספר אופטימיזציות כדוגמת: האם אנו בתוך ענן או מחוץ לענן, האם אנחנו בגובה שבו יכולים להיות עננים, האם אנו קרובים למצלמה או רחוקים ממנה, וכו'.

השוני מהאלגוריתם המוכר לנו הוא שבאלגוריתם המוכר מחליטים על אורך הצעד בצורה שונה.

באלגוריתם המוכר מתקדמים בוקטור מסוים, אך באותו הזמן נפתח רדיוס סביבנו אשר מחפש את העצם הראשון שנתקעים בו, כאשר נתקעים במשהו, הצעד הבא שלנו ימשיך מהיקף המעגל שנוצר. להלן המחשה: