תמונה שמכילה לוגו, טקסט, גופן, סימן מסחרי

התיאור נוצר באופן אוטומטי

תיכון שוהם

מנוע גרפי שיתופי

ירדן פרנק

215633686

מנחה – דוד שבת

הגנת סייבר ומערכות הפעלה

1.5.2024

תוכן עניינים

(ייתכנו עוד תתי פרקים שאינם כתובים במפורש)

[מבוא 4](#_Toc165495474)

[ייזום: 4](#_Toc165495475)

[הטכנולוגיה: 4](#_Toc165495476)

[תיחום הפרויקט: 4](#_Toc165495477)

[פירוט: 4](#_Toc165495478)

[אופן פעולת התוכנה: 4](#_Toc165495479)

[בדיקות מתוכננות: 4](#_Toc165495480)

[לו"ז: 5](#_Toc165495481)

[סיכונים: 5](#_Toc165495482)

[תיאור תחום הידע 5](#_Toc165495483)

[צד לקוח: 6](#_Toc165495484)

[צד שרת: 8](#_Toc165495485)

[מבנה / ארכיטקטורה של הפרויקט 10](#_Toc165495486)

[שרטוט: 10](#_Toc165495487)

[תיאור הטכנולוגיות: 11](#_Toc165495488)

[תיאור זרימת המידע: 12](#_Toc165495489)

[תיאור אלגוריתמים מרכזיים: 12](#_Toc165495490)

[איך מציגים אובייקטים: 12](#_Toc165495491)

[לקיחת קלט מורכב מהמשתמש: 13](#_Toc165495492)

[תיאור סביבת הפיתוח: 14](#_Toc165495493)

[תיאור פרוטוקול התקשורת: 15](#_Toc165495494)

[תיאור מסכי המערכת: 16](#_Toc165495495)

[תיאור מבנה הנתונים: 17](#_Toc165495496)

[סקירת חולשות ואיומים: 18](#_Toc165495497)

[מתקפת DDOS/DOS: 18](#_Toc165495498)

[אבטחת מידע - הצפנה: 19](#_Toc165495499)

[מימוש הפרויקט 20](#_Toc165495500)

[סקירת המחלקות/מודולים: 20](#_Toc165495501)

[קטעי קוד: 30](#_Toc165495502)

[בדיקות – Testing: 33](#_Toc165495503)

[**מדריך למשתמש...................................................................................................34**](#_Toc165495504)

[סיכום אישי / רפלקציה 35](#_Toc165495505)

[רשימה ביבליוגרפית 36](#_Toc165495506)

[נספחים 38](#_Toc165495507)

Server 38

Slient 40

Protocol 43

FileHandler 44

DiffieHellman 45

AES\_Encryption 46

ActionBar 47

Button 48

Camera 50

MatrixFunctions 52

PreObject3D 53

Projection 56

Renderer 56

Shapes [59](#_Toc165495502)

מבוא

**ייזום:**

נושא לפרויקט – תוכנת תכנון מוצר משותפת התוכנה תיתן מענה לצוות שרוצה לתכנן מוצר/מודל תלת מימדי, וצריך פלטפורמה משותפת. התוכנה תכיל בתוכה צורות גאומטריות שונות ואפשרויות לשנותן כדי ליצור את המודל בצורה נוחה. הסיבה שבחרתי בפרויקט היא שהתעניינתי בתחום של הצגות פרויקטיביות (מהמילה projection), ורציתי להכין תוכנה היכולה להציג צורות תלת מימדיות באופן עצמאי.

הייתרון של התוכנה היא שכל המשתתפים יוכלו לעבוד על האובייקטים באותו הזמן.

### הטכנולוגיה:

תוכנות שונות ליצירת מודלים תלת מימדיים קיימות כבר בשוק (כמו blender), אבל הייתרון המשמעותי של התוכנה הזאת היא ממשק המשתמש המאפשר תקשורת נוחה של כל הצוות בזמן העבודה על הפרויקט. התחומים בהם המערכת מטפלת הם ברמת התוכנה וממשק המשתמש.

### תיחום הפרויקט:

הפרויקט מתבסס על ארכיטקטורת שרת-לקוח (מרובה משתתפים), ועל כן משתמש בThreads, וגם על תכנות מונחה עצמים. בפרויקט קיימות מחלקות שונות, ולכל אחת תפקיד משלה, בשילוב עם מערכת הקבצים של המחשב (במערכת ההפעלה windows). מידע רגיש העובר דרך הsocket מאובטח באמצעות הצפנה מסוג AES ומשלב ממשק משתמש נוח באמצעות pygame.

## **פירוט:**

### אופן פעולת התוכנה:

בהתחלה תינתן אפשרות ליצור פרויקט חדש, וניתן לצרף אנשים שונים לעבודה עליו. אובייקט 'ברירת המחדל' יהיה קוביה (כמקובל בתוכנות לעיצוב תלת מימדי), ויהיו אפשרויות שונות לביצוע טרנספורמציות על האובייקטים (הזזה, כיווץ ומתיחה, סיבוב, הוספת צורות ועוד). כל כמה אובייקטים ניתן לצרף לקבוצה ולסווג אותם לרכיב שלם, ועל כל רכיב יהיה אפשר לעבוד בנפרד. את החישובים הפנימיים והתקשורת בין הלקוחות ינהל השרת הראשי ויעבוד כמתווך בין המשתתפים.

### בדיקות מתוכננות:

בדיקות אבטחה – המידע שמועבר בין כל משתמש לשרת עובר דרך sockets ויש לוודא שהמידע אינו נגיש בשום דרך כלשהי (שימוש בהצפנות).

בדיקות פעולה – בדיקות המוודאות שכל הרכיבים עובדים בצורה נכונה, שלא ייתכנו מקרים של קריסה, ושהתוכנה עובדת כראוי.

בדיקות מקרי קצה –ניסיון להזזת קודקוד אחד על קודקוד שני, לחיצה על שני כפתורים בו זמנית וכדומה.

לצורך הבדיקות ניעזר במודול icecream.

### לו"ז:

בזמן כתיבת שורות אלה, המערכת הבסיסית המציגה אובייקטים תלת-מימדיים כבר מוכנה ועובדת כראוי. מה-10.3, אתחיל לעבוד על ממשק המשתמש בשילוב עם כתיבת שרת, לקוח ושימוש במסד נתונים. במהלך חודש אפריל אני אעצב את המערכת ואפעיל את הבדיקות המתוארות מעלה, ואתקן בעיות שונות בקוד.

### סיכונים:

אבטחת מידע:

* בעיה – במהלך התקשורת בין הלקוחות לשרת, המידע שעובר צריך להיות מאובטח.
* דרך התמודדות – הצפנת המידע. נשתמש בהצפנה מסוג AES ובהעברת מפתחות עם פרוטוקול DiffieHellman.
* בעיה – חיבור רב מדי של משתתפים או מתקפת DDOS עלולים ליצור מצב של עומס על המערכת.
* דרך התמודדות – הגבלת מספר המשתתפים המקסימלי להיות 7.

העברה שגויה של מידע:

* בעיה – ייתכן שחלק מהמידע שהועבר בsocket לא יגיע במלואו
* דרך התמודדות – לכל בקשה של הלקוח, השרת ישמור אותה בקובץ המיועד רק לבקשות, וכך אם התרחש שיבוש, הוא ינסה לשלוח את ההודעה שוב.

שגיאות בזמן ריצה:

* בעיה – ייתכנו מקרים של באגים שיגרמו לקריסת המערכת או לפעולה משובשת.
* דרך התמודדות – שימוש בdebugger המובנה ובדיקה של מקרי קצה

לדוגמה, נבדוק מה קורה אם מנסים ללחוץ על כפתור העריכה פעמיים, לפני שמצב העריכה הופעל, נבדוק מה קורה אם אנחנו מנסים לשחזר קובץ ריק, וכדומה.

**תיאור תחום הידע**

## **צד לקוח:**

* שם היכולת:הצטרפות לפרויקט

מהות: הצטרפות לפרויקט, הלקוח יוכל לערוך אובייקטים באותו הפרויקט.

אוסף יכולות נדרשות: ממשק משתמש –בדיקת תקינות, הצפנה, שליחה לשרת – גם את מצב הפרויקט הנוכחי, ממשק משתמש – עדכון הפרויקט בתיקייה וירטואלית, תקשורת עם משתמשים אחרים – על המנהל של הפרויקט לאשר את ההצטרפות.

אובייקטים נחוצים: ממשק משתמש, לקיחת קלט, הצפנה, תקשורת עם השרת, בסיס נתונים זמני – שמירה זמנית לצורך שליחה אל השרת.

* שם היכולת:הצגת אובייקטים תלת מימדיים

מהות: הגדרה מראש של הקודקודים והפאות שעליהן הם נמצאים של כל אובייקט במרחב התלת מימדי, חישוב המיקום שלהם על המסך הדו מימדי, חיבור ביניהם באמצעות קווים והצגתם על מסך המשתמש.

אוסף יכולות נדרשות: כפל מטריצות, שליחה לשרת, קבלת תשובה מהשרת, פענוח, ממשק משתמש - הצגת נקודות וקווים על המסך, עדכון המסך.

אובייקטים נחוצים:  ממשק משתמש, הצפנה/פענוח, תקשורת עם השרת, מטריצות שמכפלתן עם מטריצת הקודקודים תהיה הצגה דו מימדית - לפי עקרונות הגאומטריה הפרוייקטיבית, מחלקת אובייקט האחראית לחישובים להצגתם.

* שם היכולת:תזוזה בעולם הוירטואלי

מהות: המצלמה של המשתמש תוכל לנוע במרחב לפי הקלדה במקלדת של המשתמש. בנוסף, עם העכבר (או בהגדרה מראש) תינתן אפשרות להזזת האובייקטים עצמם. למעשה, בכל תזוזה המצלמה לא באמת "זזה", אלא הקודקודים עצמם יזוזו ותיווצר אשליה של תנועה.

אוסף יכולות נדרשות: ממשק משתמש – הצגת האובייקט, חישוב המיקום החדש של הקודקודים לאחר כל תזוזה, הצגת הקודקודים. תקשורת עם השרת – להזזת האובייקטים עצמם, שליחה לשרת, קבלת תשובה מהשרת, הצפנה/פענוח, הצגת השינויים למשתמש.

אובייקטים נחוצים: מחלקת מצלמה – מנהלת את המצב נוכחי של המצלמה (מיקום, וקטורי כיוון, תזוזה, סיבוב, מהירות), קלט מהמשתמש, הצפנה/פענוח, תקשורת עם השרת.

* שם היכולת:עריכת עצמים והוספת עצמים חדשים

מהות: קיים מאגר של עצמים מובנים בתוכנה שאיתם ניתן לבנות עצמים חדשים, לערוך אותם או לחבר אותם לאובייקט קיים באמצעות איחוד קודקודים (linking). בנוסף, התוכנה תומכת בקבצי obj  כך שיהיה ניתן להציג גם אותם.

אוסף יכולות נדרשות: הזזת קודקודים ספציפיים למיקום חדש, מתיחה/כיווץ, הזזה, סיבוב, איחוד קודקודים.

אובייקטים נחוצים: מטריצות המאפשרות טרנספורמציה של האובייקט, תמיכה באיחוד קודקודים, מחלקת כפתור – המאפשרת ללחוץ על הכפתורים ולפעול בהתאם, העברת קבצי .obj לרשימת קודקודים ופאות במרחב.

* שם היכולת:תקשורת עם שאר המשתתפים

מהות: כל המשתתפים יוכלו לעבוד בו זמנית. כל תזוזה של אובייקט, הוספת אובייקט חדש וכדומה יועברו לכל הלקוחות בו זמנית.

אוסף יכולות נדרשות: קליטת נתונים והצגתם, בדיקת תקינות, הצפנה, שליחה לשרת, קבלת תשובה מהשרת, הצפנה ופענוח, הצגת התשובה למשתמש, עבודת השרת במקביל עם משתמשים רבים (סכנה – עומס יתר)

אובייקטים נחוצים: ממשק משתמש - צ'אט מובנה, תקשורת עם השרת, בסיס נתונים (בצד שרת), הצפנה ופענוח, ניהול שרת מול המשתמשים וטיפול בעומס.

* שם היכולת:הצפנה/פענוח של מידע רגיש **– גם בצד שרת**

מהות: מידע רגיש כמו סיסמאות, שמות משתמשים ומידע על הפרויקט צריכים להיות מוצפנים. בשל כך, לפני כל שליחת מידע לשרת נצפין אותו, ואחרי קבלת מידע מהשרת נפענח אותו.

אוסף יכולות נדרשות: הצפנה ופענוח, שליחה לשרת, קבלת תשובה מהשרת.

אובייקטים נחוצים: תקשורת בין השרת והלקוח באמצעות sockets, אלגוריתם הצפנה ופענוח, החלפת מפתחות מאובטחת במקרה הצורך (לדוגמה עם שיטת דיפי-הלמן).

## **צד שרת:**

* שם היכולת:חיבור/ניתוק משתמש

מהות: השרת יחבר וינתק משתמשים חדשים. יתאפשר גם מעבר בין פרויקט לפרויקט או יציאה מהפרויקט ללא התנתקות מוחלטת.

אוסף יכולות נדרשות: ניהול וארגון מידע על המשתתפים, תקשורת עם המשתמשים, הצפנה ופענוח, התמודדות עם ניתוק.

אובייקטים נחוצים: בסיס נתונים – ארגון מידע, תקשורת עם הלקוחות באמצעות sockets, הצפנה ופענוח – בשליחת מידע.

* שם היכולת:טיפול בבקשות והודעות(ניהול תקשורת)

מהות: על השרת לטפל באופן נפרד ורציף בכל בקשות הלקוחות, ולהתמודד עם עומס ותקלות בצד הלקוח.

אוסף יכולות נדרשות: ניהול וארגון מידע על המשתתפים, תקשורת עם המשתמשים, הצפנה ופענוח, ארגון בקשות ומקרים.

אובייקטים נחוצים: בסיס נתונים – ארגון ושליפת מידע, תקשורת עם הלקוחות באמצעות sockets, טיפול מחזורי ובלתי תלוי במשתמש בכל הsockets המוכנים לקריאה/כתיבה, יכולת הצפנה ופענוח.

* שם היכולת:שמירת פרוייקטים ואחזורם

מהות: השרת ישתמש בבסיס הנתונים שברשותו על מנת לשמור ולאחזר פרויקטים שלמים באופן מוצפן.

אוסף יכולות נדרשות: תקשורת עם המשתמש, קריאה וכתיבה לבסיס הנתונים, הצפנה ופענוח, בדיקת תקינות – אין לשלוח את הפרויקט לאדם לא מורשה.

אובייקטים נחוצים: בסיס נתונים, הצפנה ופענוח, תקשורת עם הלקוחות באמצעות sockets.

**מבנה / ארכיטקטורה של הפרויקט**

## שרטוט:

**תמונה שמכילה מלבן, שרטוט, לוח כתיבה לבן, טקסט

התיאור נוצר באופן אוטומטיתמונה שמכילה מלבן, שרטוט, לוח כתיבה לבן, טקסט

התיאור נוצר באופן אוטומטי**תמונה שמכילה מלבן, שרטוט, לוח כתיבה לבן, טקסט

התיאור נוצר באופן אוטומטי

**לקוח**

**לקוח**

**לקוח**

תמונה שמכילה מלבן, עיצוב

התיאור נוצר באופן אוטומטי

**מנהל קבצים**

**שרת**

שרת – server:

תפקידו של השרת הוא לטפל בפניות של הלקוח. השרת מקבל את ההודעות של הלקוח, מעבד אותן, מחשב פלט מתאים ושולח אותו בחזרה. בשרת קיימת רשימה של כלל הלקוחות המחוברים אליו, והוא יודע איזה לקוח ספציפי שלח לו הודעה, ולמי הוא צריך להחזיר. השרת הוא היחיד בעל הסמכות להשתמש במערכת הקבצים. כלומר, כאש לקוח שולח בקשה לאחזר אובייקט מסויים השמור כקובץ .obj, השרת קורא את המידע מהקובץ ושולח הודעה מתאימה בחזרה.

לקוח – client:

הלקוח, או המשתמש של המערכת, הוא האחראי להציג את התמונה על המסך, בנוסף לשליחת הודעות. מחלקת הלקוח יורשת מהמחלקה Renderer המסוגלת להציג את כלל האובייקטים. הירושה היא חשובה במקרה זה מכיוון שבמקרה של קלט מהמשתמש, לדוגמה לחיצה על שמירת האובייקטים אצל השרת, לRenderer אין את היכולת לשלוח בקשות. בנוסף, בדומה לשרת הלקוח צריך לעבד את ההודעות שמגיעות ממנו, אבל הוא גם צריך לשלב אותם בתוך המסך.

## תיאור הטכנולוגיות:

­הפרויקט מומש בשפת התכנות Python (גרסה 3.10) עם המודולים הנוספים הבאים: pygame, numpy, numba, os, pathlib, socket, threading, math, pickle ו-Crypto.

פרוטוקול תקשורת: TCP

סביבת עבודה: PyCharm (גרסה 2022.2.2)

מערכת הפעלה: windows 10

ממשק משתמש: pygame

## תמונה שמכילה תרשים, טקסט, תוכנית, שרטוט טכני התיאור נוצר באופן אוטומטיתיאור זרימת המידע:

## תיאור אלגוריתמים מרכזיים:

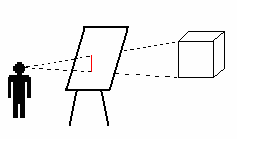
### איך מציגים אובייקטים:

כדי להציג אובייקטים תלת-מימדיים, אפשר להשתמש בהצגה שנקראת perspective projection.

פתרונות קיימים:

פתרונות אחרים כמו parallel projection (המשמרת ישרים מקבילים) או oblique projection (שבה הקרניים היוצאות מהעצם לא מאונכות למישור) יכולים גם לשמש להצגה של אובייקטים תלת מימדיים, אבל הם לא נראים מציאותיים מנקודת המבט שאנו רגילים אליה.

(לעוד סוגי הצגות תלת מימדיות - https://en.wikipedia.org/wiki/3D\_projection)



המקור של השיטה הגיע דווקא מעולם הציור. אמנים בימי הביניים החלו לשאוף לציור המגיע לדיוק מקסימלי ביחס למציאות, ולכן אימצו שיטה לציור בפרספקטיבה. הצייר ניצב מול בד ציור, ומצייר את הנראה לו מאחורי הבד. כדי לצייר עצם, יש למתוח קו ישר מהעצם אל עין הצייר, ולחתוך את הקו ישר עם מישור הבד. נשים לב שבטרנספורמציה הזאת, קווים מקבילים לא נשארים מקבילים, אלא נפגשים "באינסוף".



דוגמה לציור הממחיש הצגה פרספקטיבית:

בצורה דומה, אם נרצה לצייר עצם תלת-מימדי על המסך של המחשב שלנו, נדמיין שהמסך שלנו הוא סוג של חלון שממנו משתקף העצם. כעת, נמתח קו ישר מן העצם המשתקף על החלון ונסמן את הפיקסלים שהקו מגיע אליהם עד שנוצרת תמונה מלאה (פירוט מלא עם דוגמאות נמצא בשלב הבא: מימוש הפרויקט).

### לקיחת קלט מורכב מהמשתמש:

הצורך – נרצה שהמשתמש יוכל לשלוט ולערוך את העצמים המוצגים על המסך.

הבעיה - לספרייה pygame לא קיימת אפשרות של לחיצת כפתור.

פתרונות קיימים:

1. שימוש אך ורק במקלדת – מאוד מסורבל גם למשתמש וגם לתכנת. יגרום למספר כפתורים נחוצים מאוד גבוה.
2. שימוש בGUI אחר כמו tkinter ללקיחת קלט – יצטרך שילוב בין שתי הספריות, וגם יהיה חייב ליצור מסך חדש. <https://he.wikipedia.org/wiki/Tkinter>))

מכיוון ששתי האופציות המוצגות לעיל לא נוחות / לא פרקטיות, ניצור מחלקת כפתור משלנו.

הפרויקט דורש שני סוגים שונים של כפתורים:

1. כפתור גרירה – אם המשתמש לוחץ על הכפתור, הוא יגרר יחד עם העכבר עד לעזיבה.
2. לחיצה מיידית – אם המשתמש לוחץ על הכפתור, הוא נדלק / נכבה.

נשתמש בסוג הראשון על מנת להזיז את המיקום של האובייקטים באמצעות מערכת צירים אישית, בהזזת הקודקודים של כל אובייקט במצב עריכה ובגרירה של כפתורים ליצירת אובייקטים חדשים.

נשתמש בסוג השני לכפתורים מסוימים כמו עריכה או שמירה (מימוש מלא בחלק הבא).

## תיאור סביבת הפיתוח:

כלי הפיתוח וסביבת העבודה:

שפה – Python 3.10

סביבת עבודה – PyCharm Community 2022.2.2

GUI – pygame

בדיקות תקינות:

שימוש במודול icecream לצורך בדיקת פלט תקין

## תיאור פרוטוקול התקשורת:

התקשורת בין השרת והלקוח מתבצעת באמצעות socket מובנה בשפת python ומממש את TCP (פרוטוקול העברת שידור).

לפי הפרוטוקול, תהליך העברת המידע מתבצע באמצעות יצירת חיבור (לחיצת יד משולשת), העברת מידע וסגירת הקשר.

איך נראית הודעה:

Action: str, [data]]]

כל הודעה היא בעצם רשימה, שבה האיבר הראשון הוא הפעולה שתתבצע, והאיבר השני הוא מידע נוסף שצריך להעביר, גם הוא ברשימה. מכיוון שאי אפשר להעביר רשימה דרך socket, נשתמש בpickle המובנה בפייתון לצורך קידוד המידע למחרוזת בינארית bytes)).

דוגמה להודעה:

'save', [index + 1, (model.vertices, model.faces)]]]pickle.dumps(

כדי לאחזר את תוכן ההודעה נשתמש בpickle.loads().

|  |  |  |
| --- | --- | --- |
| Action | Data – מלקוח לשרת | תשובה – משרת ללקוח |
| 'download' | [object number] | [object number, (vertices, faces)] |
| 'alter\_all\_number' | [object number] | [Object number, new position] – שולח לכל השאר |
| 'alter\_one' | [(Object number, vertex number), new pos] | [(Object number, vertex number), new pos] – שולח לכל השאר |
| 'add' | [vertices, faces] | [vertices, faces] – שולח לכולם |
| 'save' | [Object number, (vertices, faces)] | אין |
| 'again' | אין | אין – קריאה רקורסיבית לפעולת הקידוד כי משהו לא פעל טוב |
| 'public\_key' | [user public key] | Server public key]] |

## תיאור מסכי המערכת:

מכיוון שבפרויקט הזה יש רק מסך אחד, אציג את הרכיבים השונים שקיימים בו.

חלק התצוגה:

פה מופיעים כל העצמים, מערכות הצירים והקודקודים.

בלחיצה על מערכת הצירים ניתן להזיז את האובייקט, ובלחיצה על הקודקודים (לא בתמונה – רק במצב עריכה) אפשר להזיז גם אותם.

תמונה שמכילה ספרה, עיגול, כדוק, עולם

התיאור נוצר באופן אוטומטי

סרגל אפשרויות:

תמונה שמכילה צילום מסך, נייח, אטב נייר, עיצוב

התיאור נוצר באופן אוטומטיתמונה שמכילה צילום מסך, שחור ולבן, מונוכרום, שחור

התיאור נוצר באופן אוטומטיפה נמצאים הכפתורים האחראיים לכניסה / יציאה ממצב עריכה ולשמירה של הפרוייקט.

סרגל עצמים:

גוררים את התמונה של האובייקט שאותו רוצים להוסיף, וכשעוזבים את הלחיצה, הוא מתווסף למסך.

## תיאור מבנה הנתונים:

בתוך כל תיקיית פרויקט, נמצאים:

1. תיקיית Assets המכילה את כל האובייקטים שניתן להוסיף מסרגל העצמים. בנוסף, היא גם שומרת את כל התמונות של כל הכפתורים.
2. תיקיית Objects המכילה את האובייקטים שמוצגים במסך.

(בתוך שתי התיקיות הללו העצמים נשמרים בפורמט קובץ .(obj

1. קובץ Requests - קובץ טקסט המאחסן בכל שורה את הבקשה האחרונה של כל לקוח כדי לאחזר את המידע בהצלחה במקרה שמשהו השתבש (כל שורה מיועדת רק לבקשה אחת, ולכן אם השתבש המידע בכמה בקשות, רק האחרונה תישמר).

את כל אלה, מנהל ה-FileHandler של השרת.

FileHandler היא מחלקה שמימשתי בשביל לטפל בכל העבודה עם הקבצים. ביכולה לשמור אובייקט חדש בפורמט ,.obj לאחזר אובייקט מקובץ קיים, ולעדכן את בקשות המשתמשים.

המחלקה מקבלת מהשרת את המיקום של קובץ הפרוייקט וממנו מרכיבה את המיקום של כל שאר הקבצים שהיא צריכה להשתמש בהם.

דוגמה לקובץ .obj:

תמונה שמכילה טקסט, צילום מסך, מספר, גופן

התיאור נוצר באופן אוטומטי

## סקירת חולשות ואיומים:

### מתקפת DDOS/DOS:

תיתכן בעיה אם מספר רב מדי של משתמשים ינסו להתחבר לאותו הפרויקט. לכן, הגבלתי את מספר המשתתפים המקסימלי להיות 7. אם לאחר שהצטרפו 7 לקוחות ינסה לקוח חדש להיכנס, תתקבל הודעת שגיאה:

*'number of clients should be under 7'*

### אבטחת מידע - הצפנה:

נרצה שהתשדורות שעוברות בין השרת ללקוח לא יהיו גלויים לכולם, ולכן נצטרך להצפין אותן. המידע שעובר בsocket קודם כל מוצפן לפי AES (Advanced Encryption Standard), ורק אז נשלח אל היעד. בהגיעו לשם, יתבצע קידוד ההודעה עד להגעת המידע בשלמותו.

תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטילשם ביצוע הצפנה, נדרש ששני הצדדים יחזיקו במפתח משותף המאפשר להם להצפין ולפענח הודעות. מכיוון שהלקוח לא תקשר עם השרת מלפני, יש צורך להעביר את המפתח בצורה בטוחה. הדרך שבה המפתח מועבר בפרויקט היא שימוש בפרוטוקול Diffie-Hellman

**מימוש הפרויקט**

## סקירת המחלקות/מודולים:

מחלקות מיובאות:

pygame – להצגת האובייקטים (GUI)

Numpy – למניפולציות של הקודקודים

Numba – להאצת פעולה (just in time compilation)

Os, pathlib – לעבודה עם קבצים

Socket – לתקשורת בין השרת ללקוח

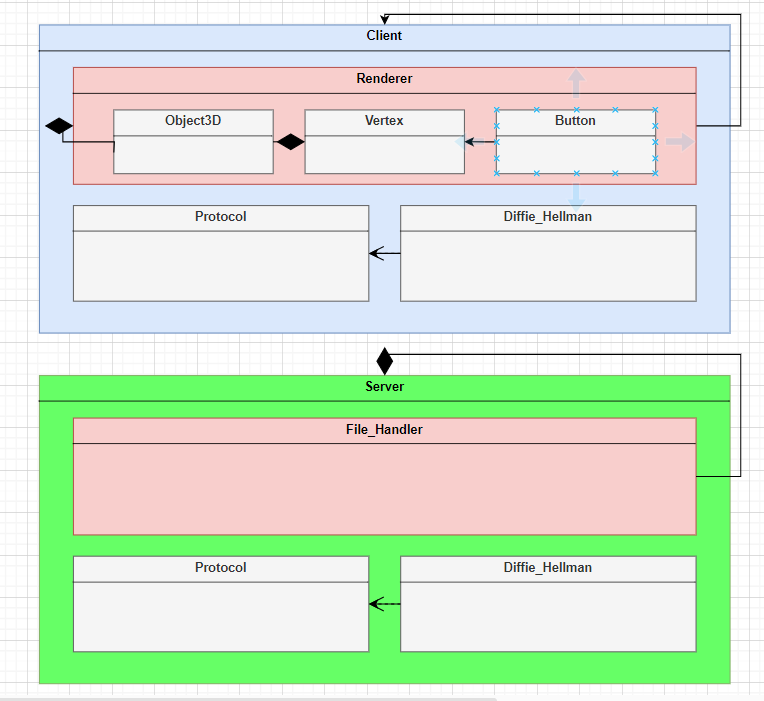
Threading – לפיצול לתהליכונים במקביל

Math – לחישובים

Pickle – לסריאליזציה / דה-סריאליזציה

Crypto.Cipher – להצפנה

**סרטוט כללי המתאר את המחלקות העיקריות, ואת היחסים ביניהם.**



**מחלקות בפיתוח עצמי:**

ㅤㅤ – צד לקוח

ㅤㅤ – צד שרת

ㅤㅤ – ממומש גם בצד שרת, וגם בצד לקוח

Button:

המחלקה יוצרת כפתור שאפשר לעבוד איתו בpygame.

|  |  |  |
| --- | --- | --- |
| שם התכונה | Type | מטרה |
| Button\_type | Str | איזה סוג |
| Screen | Pygame.display | על מי לצייר |
| Color | Tuple | צבעים למצב כבוי ודלוק |
| Image\_path | Str | איפה שמורה התמונה |
| Pos\_0 | Tuple | מיקום התחלתי |
| Image | pygame.image | תמונה |
| Rect | Pygame.Rect | המלבן שמגדיר את המיקום |
| Leave\_on | Boolean | האם להשאיר דולק כשמדליקים |
| Time\_on | int | זמן שנשאר דולק (אם רוצים לכבות אחרי כמה זמן) |
| On | Int – 1 או 0 | דולק / לא דולק |
| Drag | boolean | האם אפשר לגרור |
| Clicked | boolean | האם לחצו על הכפתור כבר |
| Let\_go | boolean | האם בדיוק הפסיקו ללחוץ |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | קלט | מטרה/פלט |
| Update | Allowed - boolean | קוראת לפעולה שמציירת את הכפתור |
| Draw | Allowed - boolean | מציירת את הכפתור, ואם allowed אמת גם בודקת אם לוחצים עליו ומזיזה אותו (אם drag אמת) |
| Drag\_button | Mouse\_state – int, allowed - boolean | מדליק \ מכבה את הכפתור אם allowed ושומרת את המיקום של העכבר אם כן |
| One\_click\_button | Mouse\_state – int, allowed | מדליקה \ מכבה את הכפתור אם allowed ומכבה אם עבר יותר מדי זמן (אםself.leave\_on שקר) |
| Turn\_on() | - | מדליקה |
| Turn\_off() | - | מכבה |
| Click() | - | מדליקה/מכבה |
| Update\_pos() | Pos - tuple | משנה מיקום |
| Rel\_update\_pos | Rel - tuple | משנה מיקום בהתאם לשינוי העכבר |
| Get\_pos() | - | מחזירה מיקום נוכחי |
| Adjust\_size() | Size - tuple, scale - int (אחד משניהם) | מגדירה תמונה, משנה את הגודל של הכפתור, צובעת אותו ומעדכנת מיקום |

ActionBar:

המחלקה מארגנת ומייצרת את כל הכפתורים שנמצאים למעלה או בצד.

|  |  |  |
| --- | --- | --- |
| שם התכונה | Type | מטרה |
| Width | Int | רוחב המלבן |
| Length | Int | אורך המלבן |
| Color | Tuple | צבע המלבן |
| Screen | Pygame.display | על מה לצייר |
| Button\_size | Tuple | גודל הכפתורים |
| Line\_color | Tuple | צבע הקווים המפרידים |
| Up\_start, side\_start | Tuple | מיקום התחלתי לשני המלבנים |
| Line\_positions\_up | List | מיקום קווים מפרידים עליונים |
| Line\_positions\_side | List | מיקום קווים מפרידים בצד |
| Bar\_buttons | List | כל הכפתורים |
| Bounding\_rect\_up | Pygame.rect | המלבן העליון |
| Bounding\_rect\_side | Pygame.rect | המלבן בצד |
| Button\_num | Int | מספר הכפתורים |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | קלט | מטרה/פלט |
| Update | - | מציירת הכל |
| Create\_buttons\_up | Screen – pygame.display | יוצרת ומחזירה את הכפתורים העליוניים |
| Create\_buttons\_side | Screen – pygame.display | יוצרת ומחזירה את הכפתורים הצדדיים |
| Set\_pos | Button\_list - list, pos[0] - tuple, length - int, side - int, line\_positions - list, button\_size – tuple | ממקמת את כל הכפתורים והקווים המפרידים במקום |

Renderer:

המחלקה אחראית על ציור של כל האובייקטים, כפתורים ועוד על המסך.

|  |  |  |
| --- | --- | --- |
| שם התכונה | Type | מטרה |
| RES | tuple | רזולוציה (אורך וגובה) |
| FPS | Int | פריימים בשנייה |
| Objects | list | כל האובייקטים |
| Object\_vertex\_alter | list | המספרים של האובייקט והקודקוד שמזיזים אותו |
| Active\_button | int | הכפתור שמזיזים כעת |
| Screen | Pygame.display | על מה לצייר |
| Bar\_button\_handler | ActionBar | אחראי על כפתורי סרגל |
| All\_buttons | List | כל הכפתורים |
| Clock | Pg.time.clock | שעון |
| Camera | Camera | מצלמה |
| Projection | Projection | לחישוב המטריצה לטרנספורציית הקודקודים |
| Ready | Pygame.rect | האם סיימנו להוריד |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | קלט | מטרה/פלט |
| Draw | - | מציירת הכל, בודקת אם במצב עריכה ומזיזה את הכפתורים |
| Get\_moving\_objects | - | מחזירה רשימה של כל האובייקטים שזזים |
| Move\_buttons | - | מזיזה את כל הכפתורים |
| Calc\_Rel | Index – int, button – Button | בודקת על איזה כפתור לוחצים ואם כן, מחזירה כמה העכבר זז |
| Move\_vertices | Rel\_pos – tuple, button - Button | מזיזה קודקודים |
| Move\_axes | Rel\_pos – tuple, button\_type - str | מזיזה ציר |
| Run | - | הפעולה המרכזית שרצה בלולאה אינסופית ומציירת את האובייקטים |

**Client:**

המחלקה משמשת לתקשורת בין המשתמש לשרת. בנוסף, היא יורשת מrenderer, כך שהיא גם מציגה את המסך (בthread נפרד).

|  |  |  |
| --- | --- | --- |
| שם התכונה | Type | מטרה |
| Objects\_num | Int | מספר אובייקטים |
| Protocol | Protocol | פרוטוקול תקשורת |
| Messages\_to\_send | List | הודעות שצריך לשלוח |
| Download\_object | Int | העצם הנוכחי שאותו מורידים |
| Saved | boolean | האם כבר שמרנו |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | קלט | מטרה/פלט |
| Receive | - | מקבלת הודעות מהשרת ופועלת בהתאם להודעה |
| Send\_data | - | בודקת את הכפתורים הרלוונטים ושולחת לשרת את ההודעה במקום 0 |
| Send\_alter\_All | - | מוסיפה להודעות אם צריך לשלוח על תזוזת אובייקט |
| Ask\_object\_to\_download | Number - int | מוסיפה הודעה לשרת על איזה אובייקט להוריד |
| Download | Data – list/tuple | מוסיפה אובייקט שהורד |
| Add | Data – list/tuple | מוסיפה אובייקט |
| Alter\_all | Object\_num – int, new\_center - tuple | מזיזה אובייקט למיקום חדש |
| Alter\_one | Object\_vertex\_number – list, vertex\_value - tuple | מזיזה מיקום של קודקוד ספציפי |
| Check\_save\_button | - | בודקת אם כפתור השמירה דלוק, ואם כן מוסיפה הודעה מתאימה |
| Check\_model\_buttons | - | בודקת אם כפתורי הוספת האובייקט דלוקים, ואם כן מוסיפה הודעה מתאימה |

**Server:**

המחלקה משמשת לתקשורת עם השרת. בנוסף, יש לה מנהל קבצים (FileHandler), שממנו היא יכולה לגשת לקבצים במערכת.

|  |  |  |
| --- | --- | --- |
| שם התכונה | Type | מטרה |
| Protocol | Protocol | פרוטוקול תקשורת |
| Client\_sockets | list | רשימת לקוחות מחוברים |
| Message\_to\_send | List | הודעה שצריך לשלוח |
| File\_handler | FileHandler | אחראי על טיפול בקבצים |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | קלט | מטרה/פלט |
| Update | - | עוברת על רשימת הלקוחות, מקבלת לקוחות חדשים ומעבירה הודעות שהם שולחים |
| Receive | Client – socket, message - list | מקבלת הודעות מלקוח ופועלת בהתאם להודעה |
| Send\_messagel | Current\_client – socket, write\_list - list | שולחת הודעה לכל הלקוחות חוץ מהנוכחי |
| Client\_joined | Client - socket | מוסיפה לקוח חדש |

**Protocol:**

מחלקה שמשתמשים בה גם השרת וגם הלקוח. היא שולחת מידע בפקטות באורך מוגדר, ואחראית על כל שליחת המידע.

|  |  |  |
| --- | --- | --- |
| שם התכונה | Type | מטרה |
| Socket | socket | Socket לשליחת הודעות |
| Max\_msg\_len | int | אורך הודעה מקסימלית |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | קלט | מטרה/פלט |
| Private\_message | Message – list, socket – socket, must\_arrive - boolean | שולחת הודעה פרטית |
| Get\_data | Socket - socket | מקודדת הודעות שמגיעות לsocket |
| Failed\_to\_receive | Socket – socket, must\_arrive - boolean | מנקה את הsocket ושולחת בקשה לשלוח שוב, אם משהו השתבש |
| Clear\_socket | Socket - socket | מנקה את הsocket |

**FileHandler:**

אחראית על ניהול הקבצים – קביעת אובייקט חדש, אחזור אובייקט קיים ושמירת בקשות המשתמשים בקובץ נפרד.

|  |  |  |
| --- | --- | --- |
| שם התכונה | Type | מטרה |
| Project\_folder | str | מיקום פרויקט |
| Objects\_folder | Str | מיקום האובייקטים |
| Assets\_folder | Str | מיקום האובייקטים האפשריים והתמונות |
| Saved | Boolean | האם כבר שמרנו |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | קלט | מטרה/פלט |
| Set\_object | Object\_number – int, data – list/tuple | שומרת אובייקט לפי הקודקודים והפאות |
| Get\_object | Object\_number – int, object\_name – str  (אחד משניהם) | מחזירה אובייקט במיקום מסוים |
| Retrieve\_objects | File\_path - str | מאחזרת אובייקט במיקום מסוים |
| New\_line\_request | - | כותבת שורה חדשה במסמך הבקשות |
| Add\_request | Request – str, index - int | מוסיפה בקשה במסמך הבקשות |
| Get\_request | Index – int | מאחזרת ומחזירה בקשה ממסמך הבקשות |
| Get\_number\_of\_objects | - | מחזירה את מספר האובייקטים שבפרויקט |

**Diffie\_Hellman:**

המחלקה מגדירה את המפתחות לפי פרוטוקול דיפי-הלמן

|  |  |  |
| --- | --- | --- |
| שם התכונה | Type | מטרה |
| Prime, base | int | מגדירים את המפתחות לפי הפרוטוקול |
| private\_key\_ | int | מפתח פרטי |
| public\_key | int | מפתח פומבי |
| שם הפעולה | קלט | מטרה/פלט |
| Generate\_keys | - | מייצרת את המפתח הפרטי והפומבי |
| Generate\_private\_key | - | מייצרת את המפתח הפרטי |
| Generate\_public\_key | - | מייצרת את המפתח הפומבי |
| Get\_private\_key,  Get\_public\_key | - | מאחזרות ומחזירות את המפתחות הפרטיים והפומביים |
| Generate\_shared\_key | Key - int | מאחזרת ומחזירה את המפתח המשותף |

**מחלקות שנעזרתי בהם במקורות חיצוניים:**

**PreObject3D:**

מחלקה קודמת ליצירת אובייקט. כל התכונות והפעולות שמתבצעות על הקודקודים, הפאות ועוד נמצאים במחלקה הזאת. היא גם זאת שמציירת את האובייקט על המסך. מכיוון שלכל אובייקט מורכב יש מערכת צירים משלו (שהיא גם אובייקט תלת מימדי), זוהי מחלקה המגדירה אובייקט בסיסי (ללא מערכת צירים) – וכיוצא מכך ללא יכולת תזוזה.

|  |  |  |
| --- | --- | --- |
| שם התכונה | Type | מטרה |
| Renderer | Renderer | האחראי לציור |
| Screen | Pygame.dispaly | על מי לצייר |
| Vertices | Np.array | רשימה של כל הקודקודים של האובייקט |
| Faces | Np.array | רשימה של כל הפאות של האובייקט |
| Draw\_vertices | Boolean | האם לצייר קודקודים |
| Active\_button | Int | הכפתור הנוכחי שלוחצים עליו |
| Object\_vertices | List | כפתורים של קודקודים – על מנת להזיז אותם |
| Fill | Boolean | האם למלא את הצורה של האובייקט |
| Axes | Axes | מערכת צירים פנימית |
| Color | Tuple | צבע rgb של האובייקט |
| Color\_faces | List | רשימה המאחדת את הפאות והצבע |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | קלט | מטרה/פלט |
| Draw | - | מציירת את האובייקט והקודקודים על המסך |
| Draw\_polygon | Vertices – np.array | מציירת את האובייקט על המסך |
| Update\_vertices | Vertices – np.array | מעדכנת את הקודקודים ומציירת אותם על המסך |
| Vertices\_to\_screen | Vertex – np.array, object\_vertex - Vertex | מציירת את הקודקודים ואת הכפתורים המתאימים להם על המסך |
| Translate | Pos - tuple | מזיזה את האובייקט למיקום חדש |
| Transform\_vertices | - | מעבירה את הקודקודים מעולם תלת מימדי לדו מימדי שאפשר להציג על המסך |
| Move\_vertex | Index – int, rel\_pos | מזיזה את הקודקודים לפי העכבר |
| Is\_moving | - | מחזירה אמת אם האובייקט נע, ושקר אם הוא לא |
| Calc\_center | - | מחזירה את הממוצע של כל הקודקודים |
| Get\_offset | Ceter1 – tuple, center2 - tuple | מחזירה רשימה של ההבדל בין שני המיקומים |

**Camera:**

מצלמה, אחראית על תזוזה על המצלמה הוירטואלית לפי הקשות המקלדת, מחשבת את המיקומים של המסך הקרוב והרחוק כדי לצייר רק את האובייקטים הנמצאים בשדה הראייה של המצלמה.

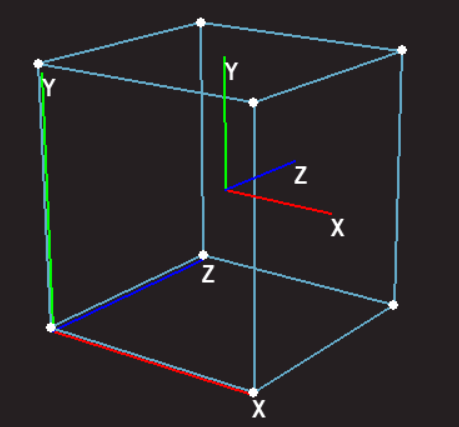
|  |  |  |
| --- | --- | --- |
| שם התכונה | Type | מטרה |
| Renderer | Renderer | האחראי לציור |
| pos | List | מיקום התחלתי |
| Reset\_pos | List | מיקום להגדרה מחדש של המצלמה |
| up | Np.array | וקטור לכיוון מעלה |
| right | Np.array | וקטור לכיוון ימינה |
| forward | Np.array | וקטור לכיוון קדימה |
| H\_fov | int | מרחק אופקי |
| v\_fov | int | מרחק אנכי |
| Near\_plane | int | מישור קרוב |
| Far\_plane | int | מישור רחוק |
| Movement\_speed | int | מהירות תנועה |
| Rotation\_speed | Int | מהירות סיבוב |
| Angle\_pitch | Int | זווית לגובה |
| Angle\_yaw | Int | זווית לצדדים |

|  |  |  |
| --- | --- | --- |
| שם הפעולה | קלט | מטרה/פלט |
| update | - | פעולה ראשית לכל פעולות המצלמה |
| Move | - | מקבלת את לחיצות המקלדת, ומזיזה את המצלמה בהתאם |
| Translate\_matrix | - | מחזירה את מטריצת ההזזה של המצלמה |
| Rotate\_matrix | - | מחזירה את מטריצת הסיבוב של המצלמה |
| Camera\_yaw | Angle - int | מסובבת את המצלמה לצדדים לפי הזווית שניתנה |
| Camera\_pitch | Angle - int | מסובבת את המצלמה למעלה ולמטה לפי הזווית שניתנה |
| Reset\_all | - | מאפסת את המיקום והסיבוב |
| Is\_reset | - | מחזירה אמת אם המצלמה אופסה, ושקר אם לא |
| Update\_axis | - | מעדכנת את וקטורי המיקום ומאפסת אותם |
| Camera\_matrix | - | מחזירה את הוקטור המעודכן של המצלמה |

## קטעי קוד:

יצירת אובייקט תלת מימדי:

1. ראשית, נגדיר את הסצנה שאנחנו רוצים לצייר במערכת צירים תלת מימדית:



לצורך הדוגמה, נניח שאורך חצי מהצלע של הקובייה הוא 1. ונסדר את הקודקודים ברשימה:

**קודקוד**

תמונה שמכילה גופן, צילום מסך, טקסט, טיפוגרפיה

התיאור נוצר באופן אוטומטי

**x y z**

נוסיף גם ערך רביעי – w=1 לכל אחד מהקודקודים. הוספה זאת נקראת קואורדינטות הומוגניות, והיא הכרחית להמשך.

בנוסף, נסדר את כל הפאות של האובייקט ברשימה, ככה שכל איבר הוא חיבור של כמה קודקודים, לפי האינדקסים של הרשימה לעיל:

**קודקודים 0, 1, 2, 3 יוצרים פאה**

**0ת 1ת 2ת 3**



1. נשנה את הקודקודים לפי רצון המשתמש. נשתמש בכפל מטריצות על מנת להזיז, לסובב, להגדיל, ולהקטין את האובייקט.

תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

הפעולה הבעייתית ביותר היא פעולת ההזזה ((translation. למעשה, לא קיימת מטריצה תלת מימדית שאם נכפיל אותה בקודקודים נקבל את הקודקודים של אובייקט זהה שהוזז לכיוון כלשהו. אולם אם נבצע טרנספורמציה אחרת שנקראת shearing שמותחת את האובייקט, רק בארבעה מימדים, נקבל בדיוק את מה שרצינו. זאת הסיבה שצריך להוסיף את הערך w=1.

1. גם בשלב הזה נשתמש בכפל מטריצות. כל רשימה של קודקודים נהפוך למערך מסוג numpy.array**,** וככה נוכל לכפול אותו במטריצות מוכנות מראש, שכל פעם עושות טרנספורמציה לקודקודים, עד שמגיעות לעולם הדו מימדי. זהו השלב הכי חשוב בכל התוכנית מכיוון שבו אנחנו בעצם מקבלים את הקואורדינטות הדו מימדיות שבהם המסך של משתמש (x ו-y).

תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

1. נציג את האובייקט המוגמר על מסך המשתמש.לשם כך, נשתמש בpygame.

נעבור על הרשימה של הפאות שלנו (ונבדוק שהקודקודים לא יצאו מהמסך), ונשתמש בפעולה pygame.draw.polygon() על מנת לצייר פאה. בנוסף, נצייר קודקודים באמצעות pygame.draw.circle().

יצירת כפתור המתאים ל-pygame:

קטעי הקוד הבאים מדגימים חלק מהפעולות שנמצאות במחלקת Button.

הפעולה draw מציגה קודם את התמונה, ומעבירה את המצב של העכבר (לחוץ או לא) ומעבירה את המידע בהתאם לסוג הכפתור (כפתור המיועד לגרירה - drag או ללחיצה חד-פעמית – one click).

תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, צילום מסך, גופן

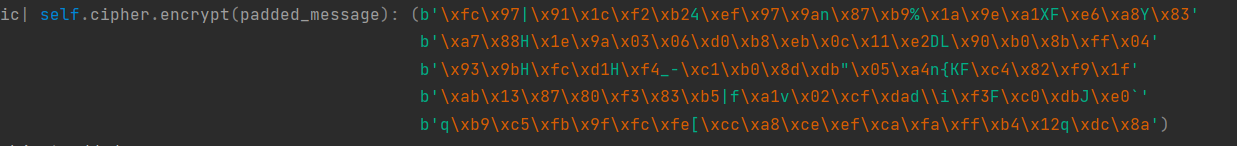
התיאור נוצר באופן אוטומטי

## בדיקות – Testing:

נייבא את המודול icecream לצורך הבדיקות.

* אבטחה:

1. בדיקות הצפנה:



נראה כי אכן תוכן ההודעה הוצפן כהלכה.

1. הגנה מפני תקיפת DDOS:

תמונה שמכילה צילום מסך, גופן, טקסט

התיאור נוצר באופן אוטומטיננסה לחבר יותר מ-7 משתתפים -

החיבור אצל הלקוח התנתק, והשרת ממשיך לפעול.

* תמונה שמכילה צילום מסך, עיגול

  התיאור נוצר באופן אוטומטיבדיקות פעולה:

המערכת עובדת, שומרת קבצים, מוסיפה אובייקטים והכפתורים עובדים.

* בדיקות מקרי קצה:

תמונה שמכילה קו, צילום מסך, צבעוני, חלל

התיאור נוצר באופן אוטומטי

לחיצה על שני כפתורים הנמצאים אחד מעל השני לא עושה בעיות.

בהתחלה, כאשר כפתור הוזז מעל כפתור אחר, השליטה עברה לכפתור האחר.

איך התמודדתי - דרך הפתרון היא סידור הכפתורים ברשימה, והקצאת משתנה שישמור לאיזה כפתור יש שליטה בכל רגע, עד לעזיבתו. חשוב שהרשימה תהיה מחוץ למחלקה כמו האובייקט לדוגמה, מכיוון שאז השליטה כן תעבור לכפתורים כמו כפתור השמירה או העריכה.

**מדריך למשתמש:**

* סביבה נדרשת: מערכת הפעלה windows, תוכנה המסוגלת להריץ python 3.
* כלים נוספים: המודולים pygame, numpy, numba, os, pathlib, socket, threading, math, pickle ו-Crypto.
* קבצי המערכת:

לאחר פתיחת התיקייה הראשונית, הנקראת 3D\_Visual, תיפתח הרשימה הבאה:

תמונה שמכילה טקסט, צילום מסך, גופן, עיצוב

התיאור נוצר באופן אוטומטי

**איך להריץ:**

בלחיצה כפולה על run, ירוצו השרת והלקוח הראשון. בלחיצה כפולה על run client 2, יתווסף לקוח חדש. (אם רוצים להריץ מתוך סביבת הפיתוח ניתן להריץ את הקבצים main ו(main2. אין להריץ לקוח חדש לפני שהשרת רץ.

כל המחלקות של הקוד נמצאות בתוך התיקייה Client\_Server.

הקבצים האחראיים להצגה התלת מימדית נמצאים בתיקייה All\_3D, קבצי הפרויקט השונים, ובהם מאוחסנים אובייקטים נמצאים בתיקיות משלהם (Project1), וכל האובייקטים האפשריים והתמונות מאוחסנים בתיקייה Assets. כל השאר נמצאים בחוץ.

איך להשתמש במערכת:

לאחר שמצליחים להפעיל את השרת והלקוח, ניתן ללחוץ על כפתור העריכה וכפתור השמירה מצד ימין למעלה. מרגע הלחיצה על כפתור העריכה, ניתן גם להזיז את הקודקודים של האובייקטים.

לחיצה על הכפתורים הצדדיים (הכפתורים עם התמונה של האובייקט), יתווסף האובייקט שלחצתם עליו למסך.

אם רוצים להזיז אובייקט, יש לגרור את אחד הצירים במערכת הצירים שלו, והוא יזוז לכיוון של הציר.

**סיכום אישי / רפלקציה:**

התחלתי לעבוד על הפרויקט מלפני כמה חודשים מכיוון שאני מאוד אוהב מתמטיקה, ונתקלתי בנושא שנקרא "גאומטריה פרויקטיבית" (מלשון projection – להציג על משהו. לאחר שהעמקתי בנושא, גיליתי שמשתמשים בטכניקות שנלמדות בנושא בגרפיקה תלת מימדית באופן נרחב, וחשבתי על לנסות לממש מנוע גרפי משל עצמי. מכיוון שלא מצאתי קורס בנושא, נעזרתי בדוגמאות קוד של אנשים מהאינטרנט, וכך הצלחתי לשפר את הידע שלי בנושא. לא תכננתי שהפרויקט יהפוך להיות זה שאני אעבוד עליו בעבודת גמר, אבל זה נראה לי הנושא הכי מעניין שאני אוכל לעשות עליו, ולכן בחרתי בו.

הכלים שאני אקח מהפרויקט הזה בהמשך הם: למידה לאורך זמן, עבודה הכוללת גם חשיבה על ממשק משתמש, וגם על התהליכים הקורים ברקע, חקירה על נושאים חדשים ותהליך כתיבת הקוד ובדיקתו. אחד הדברים שהבנתי זה שתהליך מציאת הפתרונות לוקח הרבה זמן, ולא תמיד יגיע הפתרון הספציפי לבעיה שנתקלנו בה, בשנייה שהיא צצה. לדוגמה, אצלי בפרויקט הייתה לי בעיה של סיבוב האובייקטים, והגעתי לפתרון לאחר שצפיתי בסרטון ביוטיוב בלי קשר בכלל לפרויקט.

בראייה לאחור, הייתי משקיע טיפה יותר זמן בחשיבה של לוח הזמנים, מכיוון שהיו הרבה מאוד פיצ'רים שרציתי להוסיף אבל בסוף לא הספקתי. הייתי חושב מההתחלה איך לטפל בבעיה של כפתורים שעולים אחד על השני כי בהתחלה מימשתי את הפתרון בצורה שגויה, מה שעשה לי הרבה בעיות בהמשך.

דרכים לשיפור הפרויקט, בהינתן עוד משאבים:

1. ממשק משתמש יותר ידידותי, שנותן יותר אפשרויות למשתמש, כמו בחירה אם לשמור את קבצי הפרויקט אצל השרת או אצל הלקוח, ואופציה להגדיל מסך.
2. חלוקת העומס על השרת באמצעות כמה שרתים היושבים במקומות שונים. בדרך זאת יהיה אפשר לחבר מספר רב יותר של משתמשים, וזמן התגובה היה מתקצר (אם יש שרת שיושב קרוב)
3. יותר אפשרויות לעריכה – כרגע אפשר רק להזיז קודקודים, אבל אפשר לממש דרכים יותר מגוונות כמו יצירת עקומים (אפשר באמצעותbezier curves), במקום רק קווים ישרים.
4. ייעול התהליך הגרפי – הכלים שהשתמשתי בהם כדי לעבד את התמונה, ויצירת כפתורים הם מאוד איטיים וניתן לייעל אותם באופן משמעותי. למשל, במקום לצייר פאות שלמות, אפשר לחלק אותן למשולשים ולעבד כל אחד מהם בנפרד. בנוסף, עדיף להשתמש בשפת תכנות יותר מהירה מפייתון.

**רשימה ביבליוגרפית**

* הסנר, ט. (2012, 17 באוקטובר). *‏על הגיאומטריה, הפרספקטיבה והעולם שאנו רואים,* אודיסאה. <https://www.teva.co.il/globalassets/teva-il/articles-migration/teacher/odyssey/odyssey-17.pdf#page=44>
* ויצהנדלר, ל. (23 באוגוסט 2022). *המפצחים: פרוטוקול שיתוף מפתח*. <https://davidson.weizmann.ac.il/online/askexpert/%D7%94%D7%9E%D7%A4%D7%A6%D7%97%D7%99%D7%9D-%D7%A4%D7%A8%D7%95%D7%98%D7%95%D7%A7%D7%95%D7%9C-%D7%A9%D7%99%D7%AA%D7%95%D7%A3-%D7%9E%D7%A4%D7%AA%D7%97>
* טיילור, ש. *(19 ביוני 2021) המדריך השלם לתקני הצפנה מתקדמים (AES).* [*https://he.wizcase.com/blog/%D7%94%D7%9E%D7%93%D7%A8%D7%99%D7%9A-%D7%94%D7%A9%D7%9C%D7%9D-%D7%9C%D7%AA%D7%A7%D7%A0%D7%99-%D7%94%D7%A6%D7%A4%D7%A0%D7%94-%D7%9E%D7%AA%D7%A7%D7%93%D7%9E%D7%99%D7%9D-aes/*](https://he.wizcase.com/blog/%D7%94%D7%9E%D7%93%D7%A8%D7%99%D7%9A-%D7%94%D7%A9%D7%9C%D7%9D-%D7%9C%D7%AA%D7%A7%D7%A0%D7%99-%D7%94%D7%A6%D7%A4%D7%A0%D7%94-%D7%9E%D7%AA%D7%A7%D7%93%D7%9E%D7%99%D7%9D-aes/)
* .Ilya (2021, 12 באוקטובר). *פייתון 24 - מקביליות ואסינכרוניות חלק א. Simplycode.* [*https://www.simplycode.co.il/2021/12/10/python-24-threading-asyncio-one-he/he/*](https://www.simplycode.co.il/2021/12/10/python-24-threading-asyncio-one-he/he/)
* Galea, B. (2021, June 9). *The Math behind (most) 3D games - Perspective Projection* [Video]. <https://www.youtube.com/watch?v=U0_ONQQ5ZNM>
* University of Nevada, Reno. (n.d). *The Geometry of Perspective Projection*. <https://www.cse.unr.edu/~bebis/CS791E/Notes/PerspectiveProjection.pdf>
* Wolfram MathWorld. (n.d). *Projective Geometry*. <https://mathworld.wolfram.com/ProjectiveGeometry.html>
* TutorialsPoint. (n.d). *Pygame – Overview*. <https://www.tutorialspoint.com/pygame/pygame_overview.htm>
* Jennings, N. (n.d). *Socket Programming in Python.* Real Python.<https://realpython.com/python-sockets/>

**נספחים**

**קוד הפרויקט:**

Server:

import socket  
  
import select  
from Client\_Server.FileHandler import FileHandler  
from Client\_Server.Protocol import Protocol  
from Client\_Server.DiffieHellman import DiffieHellman  
from os import listdir  
  
  
MAX\_MSG\_LENGTH = 10000  
SERVER\_PORT = 5544  
SERVER\_IP = '127.0.0.2'  
PROJECT\_FOLDER = r'Client\_Server\Project1\\'  
OBJECT\_FOLDER = PROJECT\_FOLDER + 'Objects\\'  
objects\_num = len(listdir(OBJECT\_FOLDER))  
  
# Diffie-Hellman parameters  
DH\_PRIME = 23  
DH\_BASE = 5  
  
  
class Server(socket.socket):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_(socket.AF\_INET, socket.SOCK\_STREAM)  
 self.bind((SERVER\_IP, SERVER\_PORT))  
 self.listen()  
 self.protocol = Protocol(self, MAX\_MSG\_LENGTH)  
 print("Listening for clients...")  
  
 self.client\_sockets = [] # stores the client's connection  
 self.message\_to\_send = None  
 self.file\_handler = FileHandler(PROJECT\_FOLDER)  
  
 self.diffie\_hellman = DiffieHellman(DH\_PRIME, DH\_BASE)  
 self.diffie\_hellman.generate\_keys()  
  
 self.update()  
  
 def update(self):  
 *"""this is the main loop of the server"""* while True:  
 rlist, wlist, xlist = select.select([self] + self.client\_sockets, self.client\_sockets, [])  
 for index, current\_socket in enumerate(rlist):  
 if current\_socket is self: # if a client wants to connect  
 self.client\_joined(current\_socket)  
  
 else: # if client already joined  
 message = self.protocol.get\_data(current\_socket)  
 if message:  
 self.receive(current\_socket, message)  
 self.send\_message(current\_socket, wlist)  
  
 def receive(self, client, message):  
 *"""this method is the main method for receiving data from clients"""* if message[0] == 'download':  
  
 self.file\_handler.add\_request(message, self.client\_sockets.index(client))  
 message = ['download', [message[1], self.file\_handler.get\_object(message[1])]]  
 if message[1][0] == objects\_num:  
 message[1][0] = 'last'  
 self.protocol.private\_message(message, client, True)  
  
 elif message[0] == 'again':  
 last\_request = self.file\_handler.get\_request(self.client\_sockets.index(client))  
 self.receive(client, last\_request)  
  
 elif message[0] == 'public\_key':  
 self.arrange\_keys(client, message[1])  
  
 elif message[0] == 'alter\_all\_number':  
 self.message\_to\_send = ['alter\_all\_number', message[1]] # data: [object number, new pos]  
 elif message[0] == 'alter\_one':  
 self.message\_to\_send = ['alter\_one', message[1]]  
  
 elif message[0] == 'add':  
 self.add\_object(client, message[1])  
 elif message[0] == 'save':  
 self.file\_handler.set\_object(message[1][0], message[1][1])  
 print('saved')  
  
 def send\_message(self, current\_client, write\_list):  
 *"""sends a message for every client, except for the current one"""* if self.message\_to\_send:  
 for client in write\_list:  
 if client is not current\_client:  
 self.protocol.private\_message(self.message\_to\_send, client)  
 self.message\_to\_send = None  
  
 def add\_object(self, client, data):  
 *"""adds an object on the scene"""* object\_data = self.file\_handler.get\_object(object\_name=data)  
 object\_number = self.file\_handler.get\_number\_of\_objects()  
 message = ['add', [object\_number, object\_data]] # data: [vertices, faces]  
 self.message\_to\_send = message  
 self.file\_handler.set\_object(object\_number + 1, object\_data)  
 self.protocol.private\_message(message, client)  
  
 def arrange\_keys(self, client, client\_public\_key):  
 *"""arranges shared key, using the Diffie-Hellman protocol, and sends the client the public key"""* self.protocol.private\_message(['public\_key', self.diffie\_hellman.get\_public\_key()], client)  
 shared\_key = self.diffie\_hellman.generate\_shared\_key(client\_public\_key)  
 self.protocol.set\_key(shared\_key)  
  
 def client\_joined(self, client):  
 *"""adds client to the client list"""* connection, client\_address = client.accept()  
  
 if len(self.client\_sockets) < 7:  
 print("New client joined!", client\_address)  
 self.client\_sockets.append(connection)  
 self.file\_handler.new\_line\_request()  
  
 else:  
 message = ['sorry, the server is full']  
 self.protocol.private\_message(message, connection)  
 connection.close()

:Client

import socket  
import time  
from threading import Thread  
from Client\_Server.All\_3D.Renderer import Renderer  
from Client\_Server.All\_3D.Shapes import \*  
from Client\_Server.Protocol import Protocol  
from pathlib import Path  
from Client\_Server.DiffieHellman import DiffieHellman  
  
  
MAX\_MSG\_LENGTH = 10004  
SERVER\_PORT = 5544  
IP = '127.0.0.2'  
  
# Diffie-Hellman parameters  
DH\_PRIME = 23  
DH\_BASE = 5  
  
  
class Client(socket.socket, Renderer):  
 def \_\_init\_\_(self):  
 socket.socket.\_\_init\_\_(self, socket.AF\_INET, socket.SOCK\_STREAM)  
 Thread(target=lambda: Renderer.\_\_init\_\_(self)).start()  
 self.connect((IP, SERVER\_PORT))  
 self.protocol = Protocol(self, MAX\_MSG\_LENGTH)  
 self.diffie\_hellman = DiffieHellman(DH\_PRIME, DH\_BASE)  
 self.diffie\_hellman.generate\_keys()  
 self.protocol.client\_private\_message(['public\_key', self.diffie\_hellman.get\_public\_key()])  
  
 self.messages\_to\_send = []  
 print('connected to server.')  
  
 self.download\_object = 1  
 time.sleep(0.7)  
  
 thread = Thread(target=self.receive)  
 thread.start()  
 self.send\_data()  
  
 def receive(self):  
 while True:  
 message = None  
 while not message:  
 message = self.protocol.client\_get\_data() # message: [action, [actual data]]  
 if message[0] == 'download': # ['download', [vertices, faces]]  
 if message[1][0] == 'last':  
 self.download(message[1][1], True)  
 elif self.download\_object == message[1][0]:  
 self.download(message[1][1])  
 elif message[0] == 'alter\_all\_number': # ['alter\_all\_number', [object number, new position]]  
 self.alter\_all(message[1][0], message[1][1])  
 elif message[0] == 'alter\_one':  
 self.alter\_one(message[1])  
 elif message[0] == 'public\_key':  
 shared\_key = self.diffie\_hellman.generate\_shared\_key(message[1])  
 self.protocol.set\_key(shared\_key)  
 self.ask\_object\_to\_download(1)  
 print('downloading objects...')  
 elif message[0] == 'add':  
 self.add(message[1])  
  
 def send\_data(self):  
 while True:  
 self.check\_edit\_button()  
 self.check\_save\_button()  
 self.check\_model\_buttons()  
 self.send\_alter\_all()  
  
 if len(self.messages\_to\_send) > 0:  
 if self.messages\_to\_send[0][0] == 'important':  
 self.protocol.client\_private\_message(self.messages\_to\_send[0][1:], True)  
 else:  
 self.protocol.client\_private\_message(self.messages\_to\_send[0])  
 self.messages\_to\_send = self.messages\_to\_send[1:]  
  
 def send\_alter\_all(self):  
 try:  
 print('', end='')  
 for model\_index in self.get\_moving\_objects():  
 message = ['alter\_all\_number', [model\_index, self.objects[model\_index].axes.calc\_center()]]  
 self.messages\_to\_send.append(message)  
 if self.object\_vertex\_alter:  
 self.messages\_to\_send.append(['alter\_one', self.object\_vertex\_alter])  
 except AttributeError:  
 pass  
  
 def ask\_object\_to\_download(self, number):  
 self.messages\_to\_send.append(['download', number])  
  
 def download(self, data, last=False):  
 if not self.ready:  
 new\_object = Object3D(self, data[0], data[1], number=self.download\_object)  
 self.all\_buttons += new\_object.get\_all\_buttons()  
 self.objects.append(new\_object)  
 if not last:  
 self.download\_object += 1  
 self.ask\_object\_to\_download(self.download\_object)  
 else:  
 print('finished downloading')  
 self.ready = True  
  
 def add(self, data):  
 if data[0] == self.download\_object:  
 self.download\_object += 1  
 new\_object = Object3D(self, data[1][0], data[1][1], number=self.download\_object)  
 self.objects.append(new\_object)  
 self.all\_buttons += new\_object.get\_all\_buttons()  
 print('object added')  
  
 def alter\_all(self, object\_num, new\_center):  
 try:  
 model = self.objects[object\_num]  
 axes = model.axes  
 axes.translate(axes.get\_offset(new\_center, axes.calc\_center()))  
 except IndexError:  
 pass  
  
 def alter\_one(self, object\_vertex\_alter):  
 object\_number, vertex\_number, new\_pos = object\_vertex\_alter  
 if len(self.objects) > object\_number:  
 self.objects[object\_number].vertices[vertex\_number] = new\_pos  
  
 def check\_edit\_button(self):  
 if self.bar\_button\_handler.bar\_buttons[0].on: # edit, save button in client module  
 self.edit = True  
 for model in self.objects[1:]:  
 model.draw\_vertices = True  
 elif self.edit:  
 self.edit = False  
 for model in self.objects[1:]:  
 model.draw\_vertices = False  
  
 def check\_save\_button(self):  
 try:  
 button = self.bar\_button\_handler.bar\_buttons[1]  
 if button.let\_go: # save project  
 for index, model in enumerate(self.objects[1:]):  
 self.messages\_to\_send.append(['important', 'save', [index + 1, (model.vertices, model.faces)]])  
 button.let\_go = False  
 except AttributeError:  
 pass  
  
 def check\_model\_buttons(self):  
 for button in self.bar\_button\_handler.bar\_buttons[2:]:  
 if button.let\_go:  
 model\_name = Path(button.image\_path).stem  
 self.messages\_to\_send.append(['important', 'add', model\_name])  
 button.update\_pos(button.pos\_0)  
 button.let\_go = False

Protocol:

import pickle  
from Client\_Server.AES\_Encryption import AESCipher  
  
  
class Protocol:  
 def \_\_init\_\_(self, socket, max\_message\_length,):  
 self.socket = socket  
 self.max\_msg\_len = max\_message\_length  
 self.AES\_cipher = None  
  
 def private\_message(self, message, socket, must\_arrive=False):  
 message = pickle.dumps(message)  
 if self.AES\_cipher:  
 message = self.AES\_cipher.encrypt\_message(message)  
  
 index, message\_length = 0, len(message)  
 if must\_arrive:  
 if message\_length <= self.max\_msg\_len:  
 socket.send(b'im' + message)  
 else:  
 socket.send(b'im' + b'on' + message[:self.max\_msg\_len - 4])  
 index = self.max\_msg\_len - 4  
  
 while index < message\_length - self.max\_msg\_len:  
 new\_message = b'on' + message[index:index + self.max\_msg\_len - 4]  
 socket.send(new\_message)  
 index += self.max\_msg\_len - 4  
 socket.send(message[index:])  
  
 def get\_data(self, socket):  
 data\_received = []  
 message = socket.recv(self.max\_msg\_len)  
 must\_arrive = message[:2] == b'im'  
 if must\_arrive:  
 message = message[2:]  
  
 while message[:2] == b'on':  
 data\_received.append(message[2:])  
 message = socket.recv(self.max\_msg\_len)  
 data\_received.append(message)  
 self.clear\_socket(socket)  
  
 try:  
 complete\_message = b''.join(data\_received)  
 if self.AES\_cipher:  
 complete\_message = self.AES\_cipher.decrypt\_message(complete\_message)  
 return pickle.loads(complete\_message)  
 except Exception as e:  
 print(f'Error: {e}, trying again')  
 return self.failed\_to\_receive(socket, must\_arrive)  
  
 def failed\_to\_receive(self, socket, must\_arrive=False):  
 self.clear\_socket(socket)  
 if must\_arrive:  
 socket.send(pickle.dumps(['again']))  
 return None  
  
 def clear\_socket(self, socket):  
 socket.setblocking(False)  
 try:  
 while True:  
 socket.recv(self.max\_msg\_len)  
 except BlockingIOError:  
 self.socket.setblocking(True)  
  
 def client\_private\_message(self, message, must\_arrive=False):  
 self.private\_message(message, self.socket, must\_arrive)  
  
 def client\_clear\_socket(self):  
 self.clear\_socket(self.socket)  
  
 def client\_get\_data(self):  
 return self.get\_data(self.socket)  
  
 def set\_key(self, key):  
 if not self.AES\_cipher:  
 self.AES\_cipher = AESCipher(key)  
 else:  
 self.AES\_cipher.set\_key(key)

FileHandler:

from os import listdir  
  
  
class FileHandler:  
 def \_\_init\_\_(self, project\_folder):  
 self.project\_folder = project\_folder  
 self.objects\_folder = project\_folder + 'Objects\\'  
 self.assets\_folder = project\_folder + 'Assets\\'  
 self.requests\_path = project\_folder + 'Requests.txt'  
 open(self.requests\_path, 'w').close()  
  
 def set\_object(self, object\_number, data):  
 vertices, faces = data  
 file\_path = f'{self.objects\_folder}object {object\_number}.obj'  
  
 with open(file\_path, 'w') as object\_file:  
 for vertex in vertices:  
 vertex = [str(value) for value in vertex]  
 object\_file.write(f'v {" ".join(vertex[:-1])}\n')  
 for face in faces:  
 face = [str(value + 1) for value in face]  
 object\_file.write(f'f {" ".join(face)}\n')  
  
 def get\_object(self, object\_number=None, object\_name=None):  
 if object\_number:  
 path = self.objects\_folder  
 name = f'object {object\_number}.obj'  
 else:  
 path = self.assets\_folder  
 name = object\_name + '.obj'  
 return self.retrieve\_object(path + name)  
  
 @staticmethod  
 def retrieve\_object(file\_path):  
 vertices, faces = [], []  
  
 with open(file\_path, 'r') as object\_file:  
 for line in object\_file:  
 if line.startswith('v'):  
 vertices.append([float(i) for i in line.split()[1:]] + [1])  
 elif line.startswith('f'):  
 faces\_ = line.split()[1:]  
 faces.append([int(face\_.split('/')[0]) - 1 for face\_ in faces\_])  
 return vertices, faces  
  
 def new\_line\_request(self):  
 with open(self.requests\_path, 'a') as requests\_file:  
 requests\_file.write('\n ')  
  
 def add\_request(self, request, index):  
 with open(self.requests\_path, 'r') as requests\_file:  
 lines = requests\_file.readlines()  
 lines[index] = ' '.join(str(element) for element in request)  
 with open(self.requests\_path, 'w') as requests\_file:  
 requests\_file.writelines(lines)  
  
 def get\_request(self, index):  
 with open(self.requests\_path, 'r') as requests\_file:  
 lines = requests\_file.readlines()  
 request = list(lines[index].split(' '))  
 return request  
  
 def get\_number\_of\_objects(self):  
 return len(listdir(self.objects\_folder))

DiffieHellman:

import random  
  
  
class DiffieHellman:  
 def \_\_init\_\_(self, prime, base):  
 if type(prime) is not int or prime % 2 == 0:  
 raise ValueError('prime is not an integer, or is not even')  
 if type(base) is not int or not 1 < base < prime:  
 raise ValueError('base is not an integer, or is not less than prime and bigger than 1')  
  
 self.prime, self.base = prime, base  
 self.\_private\_key = 0  
 self.public\_key = 0  
  
 def generate\_keys(self):  
 self.generate\_private\_key()  
 self.generate\_public\_key()  
  
 def generate\_private\_key(self):  
 self.\_private\_key = random.randint(1, self.prime - 1)  
  
 def get\_private\_key(self):  
 return self.\_private\_key  
  
 def generate\_public\_key(self):  
 self.public\_key = pow(self.base, self.\_private\_key) % self.prime  
  
 def get\_public\_key(self):  
 return self.public\_key  
  
 def generate\_shared\_key(self, key):  
 return (pow(key, self.\_private\_key) % self.prime).to\_bytes(16, byteorder='big')

AES\_Encryption:

from Crypto.Cipher import AES  
from Crypto.Util.Padding import pad, unpad  
  
  
class AESCipher:  
 def \_\_init\_\_(self, key):  
 self.\_key = key  
  
 def encrypt\_message(self, message):  
 if self.valid\_message(message):  
 cipher = AES.new(self.\_key, AES.MODE\_ECB)  
 padded\_message = pad(message, AES.block\_size)  
 return cipher.encrypt(padded\_message)  
 return message  
  
 def decrypt\_message(self, encrypted\_message):  
 if self.valid\_message(encrypted\_message):  
 cipher = AES.new(self.\_key, AES.MODE\_ECB)  
 return unpad(cipher.decrypt(encrypted\_message), AES.block\_size)  
 return encrypted\_message  
  
 @staticmethod  
 def valid\_message(message):  
 for code in special\_codes:  
 if code in message:  
 return False  
 return True  
  
 def get\_key(self):  
 return self.\_key  
  
 def set\_key(self, key):  
 self.\_key = key

**All\_3D:**

ActionBar:

from Client\_Server.All\_3D.Button import Button  
import pygame as pg  
IMAGES\_PATH = r'Client\_Server\Assets\Images\\'  
  
  
class ActionBar:  
 def \_\_init\_\_(self, screen, size, color):  
 self.width, self.length = size  
 self.color = color  
 self.screen = screen  
  
 self.button\_size = (50, 50)  
 self.line\_color = (105, 105, 105)  
 self.up\_start, self.side\_start = (0, 0), (self.width \* 0.91, self.length \* 2.5)  
 self.line\_positions\_up, self.line\_positions\_side = [], []  
 self.bar\_buttons = self.create\_buttons\_up(screen) + self.create\_buttons\_side(screen)  
 self.line\_positions\_side = self.line\_positions\_side[:-1]  
  
 self.bounding\_rect\_up = pg.Rect(self.up\_start, (self.width, self.length))  
 button\_num = len(self.bar\_buttons)  
 self.bounding\_rect\_side = pg.Rect(self.side\_start, (self.length, (button\_num + 1.25) \* self.button\_size[0]))  
  
 def update(self):  
 pg.draw.rect(self.screen, self.color, self.bounding\_rect\_up)  
 pg.draw.rect(self.screen, self.color, self.bounding\_rect\_side)  
 for button in self.bar\_buttons:  
 button.update(True)  
 for pos in self.line\_positions\_up:  
 pg.draw.line(self.screen, self.line\_color, (pos, 0), (pos, self.length - 1))  
 for pos in self.line\_positions\_side:  
 pg.draw.line(self.screen, self.line\_color, (self.side\_start[0], pos), (self.side\_start[0] + self.length - 1, pos))  
  
 def create\_buttons\_up(self, screen):  
 edit\_button = Button(screen, (0, 0),  
 image=IMAGES\_PATH + 'edit.png',  
 scale=0.15, buttontype='edit')  
 save\_button = Button(screen, (0, 0), image=IMAGES\_PATH + 'save.png',  
 scale=0.25, drag=True, buttontype='edit')  
 button\_list = [edit\_button, save\_button]  
 self.set\_pos(button\_list, self.up\_start, self.length, 0, self.line\_positions\_up, self.button\_size)  
 return button\_list  
  
 def create\_buttons\_side(self, screen):  
 line\_button = Button(screen, (0, 0), image=IMAGES\_PATH + 'line.png', size=self.button\_size, drag=True, buttontype='model')  
 cube\_button = Button(screen, (0, 0), image=IMAGES\_PATH + 'cube.png', size=self.button\_size, drag=True, buttontype='model')  
 pyramid\_button = Button(screen, (0, 0), image=IMAGES\_PATH + 'pyramid.png', size=self.button\_size, drag=True, buttontype='model')  
 circle\_button = Button(screen, (0, 0), image=IMAGES\_PATH + 'circle.png', size=self.button\_size, drag=True, buttontype='model')  
 sphere\_button = Button(screen, (0, 0), image=IMAGES\_PATH + 'sphere.png', size=self.button\_size, drag=True, buttontype='model')  
 deer\_button = Button(screen, (0, 0), image=IMAGES\_PATH + 'deer.png', size=self.button\_size, drag=True, buttontype='model')  
 tank\_button = Button(screen, (0, 0), image=IMAGES\_PATH + 'tank.png', size=self.button\_size, drag=True, buttontype='model')  
 button\_list = [line\_button, cube\_button, pyramid\_button, circle\_button, sphere\_button, deer\_button, tank\_button]  
 self.set\_pos(button\_list, self.side\_start, self.length, 1, self.line\_positions\_side, self.button\_size)  
 return button\_list  
  
 @staticmethod  
 def set\_pos(button\_list, pos\_0, length, side, line\_positions, button\_size): # side: 0 = up, 1 = right  
 pos = [pos\_0[0] + length // 2, pos\_0[1] + length // 2]  
 increment = button\_size[side] \* 1.5  
 for button in button\_list:  
 button.update\_pos(pos)  
 button.pos\_0 = tuple(pos)  
 line\_positions.append(pos[side] + increment // 2)  
 pos[side] += increment

Button:

import pygame as pg  
  
  
def colorize(image, newColor):  
 if not image:  
 return None  
 image = image.copy()  
 image.fill((0, 0, 0, 255), None, pg.BLEND\_RGBA\_MULT)  
 image.fill(newColor[0:3] + (0,), None, pg.BLEND\_RGBA\_ADD)  
  
 return image  
  
  
# button class  
class Button:  
 def \_\_init\_\_(self, screen, pos, size=None, image=None, scale=None, drag=False, leave\_on=True, buttontype=None):  
 self.button\_type = buttontype  
 self.screen = screen  
 self.color = (231, 231, 231), (237, 175, 2) # deactivated, activated  
 self.image\_path = image  
 self.pos\_0 = None  
 self.image, self.rect = None, None  
 self.adjust\_size(size=size, scale=scale, image=image)  
 self.rect.center = pos  
 self.leave\_on = leave\_on  
 self.time\_on = 0  
 self.on = 0  
 self.drag = drag # if the purpose of pressing the button is to drag it  
 self.clicked = False  
 self.let\_go = False  
  
 def update(self, allowed):  
 self.draw(allowed)  
  
 def draw(self, allowed):  
 # draw button on screen  
 if self.image:  
 self.screen.blit(self.image, (self.rect.x, self.rect.y))  
 mouse\_state = pg.mouse.get\_pressed()[0]  
 if self.drag:  
 self.drag\_button(mouse\_state, allowed)  
 else:  
 self.one\_click\_button(mouse\_state, allowed)  
  
 def drag\_button(self, mouse\_state, allowed):  
 if mouse\_state == 1 and not self.on and allowed:  
 pos = pg.mouse.get\_pos()  
 if self.rect.collidepoint(pos):  
 self.turn\_on()  
 elif mouse\_state == 0 and self.on:  
 self.let\_go = True  
 self.turn\_off()  
  
 def one\_click\_button(self, mouse\_state, allowed):  
 if mouse\_state == 1 and allowed and not self.clicked:  
 pos = pg.mouse.get\_pos()  
 if self.rect.collidepoint(pos):  
 self.click()  
 self.clicked = True  
 elif mouse\_state == 0:  
 self.clicked = False  
 if self.on and not self.leave\_on:  
 self.time\_on += 1  
 if self.time\_on > 7:  
 self.time\_on = 0  
 self.click()  
  
 def turn\_on(self):  
 self.on = 1  
 self.image = colorize(self.image, self.color[1])  
  
 def turn\_off(self):  
 self.on = 0  
 self.image = colorize(self.image, self.color[0])  
  
 def click(self):  
 self.on = 1 - self.on  
 self.image = colorize(self.image, self.color[self.on])  
  
 def update\_pos(self, pos):  
 self.rect.center = pos  
  
 def rel\_update\_pos(self, rel):  
 current\_pos = self.get\_pos()  
 self.rect.center = (current\_pos[0] + rel[0], current\_pos[1] + rel[1])  
  
 def get\_pos(self):  
 return self.rect.center  
  
 def adjust\_size(self, size=None, scale=None, image=None): # either size or scale  
 if image:  
 image = pg.image.load(image)  
 if scale:  
 size = image.get\_width() \* scale, image.get\_height() \* scale  
 image = pg.transform.smoothscale(image, size)  
 image = colorize(image, self.color[0])  
 self.image = image  
 self.rect = pg.Rect((0, 0), size)

Camera:

import pygame as pg  
from Client\_Server.All\_3D.MatrixFunctions import \*  
  
  
class Camera:  
 def \_\_init\_\_(self, renderer, pos):  
 self.renderer = renderer  
 self.pos = pos # camera position  
 self.reset\_pos = [0.001, 2, -9, 1]  
  
 self.up = np.array([0, 1, 0, 1]) # vectors that define the camera's direction  
 self.right = np.array([1, 0, 0, 1])  
 self.forward = np.array([0, 0, 1, 1])  
  
 self.h\_fov = math.pi / 3 # horizontal and vertical fields of view  
 self.v\_fov = self.h\_fov \* (renderer.HEIGHT / renderer.WIDTH)  
  
 self.near\_plane = 0.1 # clipping planes for the camera  
 self.far\_plane = 100  
  
 self.movement\_speed = 0.07  
 self.rotation\_speed = 0.015  
  
 self.angle\_pitch = 0  
 self.angle\_yaw = 0  
  
 def update(self):  
 self.move()  
 self.update\_axis()  
  
 def move(self):  
 key = pg.key.get\_pressed()  
 if key[pg.K\_w]: # movement  
 self.pos += self.movement\_speed \* self.forward  
 if key[pg.K\_s]:  
 self.pos -= self.movement\_speed \* self.forward  
 if key[pg.K\_d]:  
 self.pos += self.movement\_speed \* self.right  
 if key[pg.K\_a]:  
 self.pos -= self.movement\_speed \* self.right  
 if key[pg.K\_q]:  
 self.pos += self.movement\_speed \* self.up  
 if key[pg.K\_e]:  
 self.pos -= self.movement\_speed \* self.up  
  
 if key[pg.K\_LEFT]:  
 self.camera\_yaw(-self.rotation\_speed)  
 if key[pg.K\_RIGHT]:  
 self.camera\_yaw(self.rotation\_speed)  
 if key[pg.K\_UP]:  
 if self.angle\_pitch > -1.5:  
 self.camera\_pitch(-self.rotation\_speed)  
 if key[pg.K\_DOWN]:  
 if self.angle\_pitch < 1.5:  
 self.camera\_pitch(self.rotation\_speed)  
  
 if key[pg.K\_r]: # reset  
 self.reset\_all()  
  
 def translate\_matrix(self):  
 x, y, z, w = self.pos  
 return np.array([  
 [1, 0, 0, 0],  
 [0, 1, 0, 0],  
 [0, 0, 1, 0],  
 [-x, -y, -z, 1]  
 ])  
  
 def rotate\_matrix(self):  
 rx, ry, rz, w = self.right  
 fx, fy, fz, w = self.forward  
 ux, uy, uz, w = self.up  
 return np.array([  
 [rx, ux, fx, 0],  
 [ry, uy, fy, 0],  
 [rz, uz, fz, 0],  
 [0, 0, 0, 1]  
 ])  
  
 def camera\_yaw(self, angle):  
 self.angle\_yaw += angle  
  
 def camera\_pitch(self, angle):  
 self.angle\_pitch += angle  
  
 def reset\_axis(self):  
 self.up = np.array([0, 1, 0, 1])  
 self.right = np.array([1, 0, 0, 1])  
 self.forward = np.array([0, 0, 1, 1])  
  
 def reset\_all(self):  
 self.pos = self.reset\_pos  
 self.reset\_axis()  
 self.angle\_pitch = 0  
 self.angle\_yaw = 0  
  
 def is\_reset(self):  
 return list(self.pos) == self.reset\_pos and self.angle\_yaw == 0 and self.angle\_pitch == 0  
  
 def update\_axis(self):  
 self.reset\_axis()  
 rotate = rotate\_x(self.angle\_pitch) @ rotate\_y(self.angle\_yaw)  
  
 self.forward = self.forward @ rotate  
 self.right = self.right @ rotate  
 self.up = self.up @ rotate  
  
 def camera\_matrix(self):  
 # the product of the two matrices above gives the matrix that will allow to make  
 # the world's coordinates coincide with the camera's coordinates  
 return self.translate\_matrix() @ self.rotate\_matrix()

MatrixFunctions:

import math  
import numpy as np  
  
  
def translate(pos):  
 tx, ty, tz = pos  
 return np.array([  
 [1, 0, 0, 0],  
 [0, 1, 0, 0],  
 [0, 0, 1, 0],  
 [tx, ty, tz, 1]  
 ])  
  
  
def rotate\_x(a):  
 return np.array([  
 [1, 0, 0, 0],  
 [0, math.cos(a), math.sin(a), 0],  
 [0, -math.sin(a), math.cos(a), 0],  
 [0, 0, 0, 1]  
 ])  
  
  
def rotate\_y(a):  
 return np.array([  
 [math.cos(a), 0, -math.sin(a), 0],  
 [0, 1, 0, 0],  
 [math.sin(a), 0, math.cos(a), 0],  
 [0, 0, 0, 1]  
 ])  
  
  
def rotate\_z(a):  
 return np.array([  
 [math.cos(a), math.sin(a), 0, 0],  
 [-math.sin(a), math.cos(a), 0, 0],  
 [0, 0, 1, 0],  
 [0, 0, 0, 1]  
 ])  
  
  
def scale(n):  
 return np.array([  
 [n, 0, 0, 0],  
 [0, n, 0, 0],  
 [0, 0, n, 0],  
 [0, 0, 0, 1]  
 ])

PreObject3D:

import pygame as pg  
from Client\_Server.All\_3D.MatrixFunctions import \*  
from Client\_Server.All\_3D.Button import Button  
import numpy as np  
from numba import njit  
  
  
@njit(fastmath=True)  
def any\_func(arr, a, b):  
 return np.any((arr == a) | (arr == b))  
  
  
CUBE\_VERTICES = [(0, 0, 0, 1), (0, 1, 0, 1), (1, 1, 0, 1), (1, 0, 0, 1), (0, 0, 1, 1), (0, 1, 1, 1), (1, 1, 1, 1), (1, 0, 1, 1)]  
CUBE\_FACES = [(0, 1, 2, 3), (4, 5, 6, 7), (0, 4, 5, 1),  
 (2, 3, 7, 6), (1, 2, 6, 5), (0, 3, 7, 4)]  
  
  
class PreObject3D:  
 def \_\_init\_\_(self, renderer, vertices=None, faces=None, filename=None, axes=None, object\_number=0):  
 self.renderer = renderer  
 self.screen = renderer.screen  
  
 if filename:  
 self.vertices, self.faces = self.get\_object\_from\_file(filename)  
 elif vertices is None or faces is None:  
 self.vertices, self.faces = np.array(CUBE\_VERTICES), np.array(CUBE\_FACES) # arrays of vertices and lines  
 else:  
 self.vertices, self.faces = vertices, faces  
  
 self.draw\_vertices = False  
 self.active\_button = None  
 self.object\_vertices = [Vertex(self.screen, (0, 0), object\_number=object\_number, number=i) for i in range(len(self.vertices))]  
  
 self.fill = False  
  
 self.axes = axes  
 self.color = [100, 170, 200]  
 self.color\_faces = [(pg.Color(self.color), face) for face in self.faces]  
 pg.font.init()  
 self.font = pg.font.SysFont('Arial', 20, bold=True)  
 self.label = ''  
  
 def draw(self):  
 *"""main function that displays the object"""* vertices = self.transform\_vertices() # transforming the vertices to display on 2D  
 self.draw\_polygon(vertices)  
 self.update\_vertices(vertices)  
  
 def draw\_polygon(self, vertices):  
 for index, color\_face in enumerate(self.color\_faces): # drawing the faces  
 color, face = color\_face  
 polygon = vertices[face] # get vertices  
 if not any\_func(polygon, self.renderer.HALF\_WIDTH, self.renderer.HALF\_HEIGHT): # if the polygon didn't exit the screen  
 if self.fill and len(polygon) >= 3: # if it's a shape that can be shaded  
 line\_thickness = 0 # (fill the polygon)  
 else:  
 line\_thickness = 2  
 pg.draw.polygon(self.screen, color, polygon, line\_thickness) # drawing the face  
  
 if self.label: # drawing a label if exists  
 text = self.font.render(self.label[index], True, 'white')  
 self.screen.blit(text, polygon[-1])  
  
 def update\_vertices(self, vertices):  
 for index, vertex in enumerate(vertices):  
 object\_vertex = self.object\_vertices[index]  
 object\_vertex.set\_pos(vertex)  
 if self.draw\_vertices:  
 self.vertices\_to\_screen(vertex, object\_vertex)  
  
 def vertices\_to\_screen(self, vertex, object\_vertex):  
 if not any\_func(vertex, self.renderer.HALF\_WIDTH, self.renderer.HALF\_HEIGHT): # if the vertex didn't exit the screen  
 object\_vertex.update(not bool(self.active\_button))  
  
 def translate(self, pos):  
 self.vertices = self.vertices @ translate(pos)  
  
 def scale(self, times):  
 self.vertices = self.vertices @ scale(times)  
  
 def rotate\_x(self, angle):  
 self.vertices = self.vertices @ rotate\_x(angle)  
  
 def rotate\_y(self, angle):  
 self.vertices = self.vertices @ rotate\_y(angle)  
  
 def rotate\_z(self, angle):  
 self.vertices = self.vertices @ rotate\_z(angle)  
  
 def transform\_vertices(self):  
 vertices = self.vertices @ self.renderer.camera.camera\_matrix() # transfer vertices to fit in the camera's coordinate system  
 vertices = vertices @ self.renderer.projection.projection\_matrix # transfer vertices to clip space  
 vertices /= vertices[:, -1].reshape(-1, 1) # normalize the value of the vertices (divide by w)  
 vertices[(vertices > 1.5) | (vertices < -1.5)] = 0 # remove te vertices that don't fit into the screen  
 vertices = vertices @ self.renderer.projection.to\_screen\_matrix # transforming the vertices to the screen's width and height  
 vertices = vertices[:, :2] # considering only the x and y coordinates  
 return vertices  
  
 @staticmethod  
 def get\_object\_from\_file(file\_path):  
 vertices, faces = [], []  
  
 with open(file\_path, 'r') as object\_file:  
 for line in object\_file:  
 if line.startswith('v '):  
 vertices.append([float(i) for i in line.split()[1:]] + [1])  
 elif line.startswith('f'):  
 faces\_ = line.split()[1:]  
 faces.append([int(face\_.split('/')[0]) - 1 for face\_ in faces\_])  
  
 return vertices, faces  
  
 def move\_vertex(self, index, rel\_pos):  
 self.vertices[index][0] += rel\_pos[0]  
 self.vertices[index][1] += rel\_pos[1]  
  
 def is\_moving(self):  
 if self.axes:  
 offset = self.get\_offset(self.axes.calc\_center(), self.calc\_center())  
 for index, value in enumerate(offset):  
 if abs(value) > 0.01:  
 return True  
 return False  
  
 def calc\_center(self):  
 return np.mean(self.vertices, axis=0)  
  
 @staticmethod  
 def get\_offset(center1, center2):  
 return list(np.subtract(center1, center2))[:3]  
  
  
class Vertex(Button):  
 def \_\_init\_\_(self, screen, pos, vertex\_size=4, number=0, object\_number=0):  
 self.vertex\_size = vertex\_size  
 super().\_\_init\_\_(screen, pos, size=(vertex\_size \* 2, vertex\_size \* 2), drag=True, buttontype=f'{object\_number}vertex{number}')  
  
 def set\_pos(self, pos):  
 self.rect.center = pos  
  
 def update(self, allowed):  
 self.draw(allowed)  
 pg.draw.circle(self.screen, 'white', self.rect.center, self.vertex\_size)

Projection:

import math  
import numpy as np  
  
  
class Projection:  
 def \_\_init\_\_(self, renderer):  
 near = renderer.camera.near\_plane  
 far = renderer.camera.far\_plane  
 right = math.tan(renderer.camera.h\_fov / 2)  
 left = -right  
 top = math.tan(renderer.camera.v\_fov / 2)  
 bottom = -top  
  
 m00 = 2 / (right - left)  
 m11 = 2 / (top - bottom)  
 m22 = (far + near) / (far - near)  
 m32 = -2 \* near \* far / (far - near)  
 self.projection\_matrix = np.array([  
 [m00, 0, 0, 0],  
 [0, m11, 0, 0],  
 [0, 0, m22, 1],  
 [0, 0, m32, 0]  
 ])  
  
 # transforming the vertices to the screen's width and height  
 hw, hh = renderer.HALF\_WIDTH, renderer.HALF\_HEIGHT  
 self.to\_screen\_matrix = np.array([  
 [hw, 0, 0, 0],  
 [0, -hh, 0, 0],  
 [0, 0, 1, 0],  
 [hw, hh, 0, 1]  
 ])

Renderer:

\_\_author\_\_ = 'Yarden Frank'  
  
from Client\_Server.All\_3D.Projection import Projection  
from Client\_Server.All\_3D.Camera import Camera  
from Client\_Server.All\_3D.Shapes import \*  
from Client\_Server.All\_3D.Action\_Bar import ActionBar  
  
  
class Renderer:  
 def \_\_init\_\_(self):  
 self.RES = self.WIDTH, self.HEIGHT = 1600 \* 0.8, 900 \* 0.8 # resolution  
 self.HALF\_WIDTH, self.HALF\_HEIGHT = self.WIDTH // 2, self.HEIGHT // 2  
 self.FPS = 60  
 self.objects = []  
 self.object\_vertex\_alter = None  
  
 self.active\_button = None # button numbers: all bar buttons, object1 buttons - axes, vertices, object2...  
 self.edit = False  
 self.screen = pg.display.set\_mode(self.RES)  
 self.bar\_button\_handler = ActionBar(self.screen, (self.WIDTH, 50), (40, 40, 40))  
 self.all\_buttons = list(self.bar\_button\_handler.bar\_buttons)[1:]  
 self.clock = pg.time.Clock()  
 self.camera = Camera(self, [5, 6, -9, 1])  
 self.camera.camera\_pitch(0.28)  
 self.camera.camera\_yaw(-0.5)  
 self.projection = Projection(self)  
 self.ready = False  
  
 self.objects.insert(0, Axes(self))  
 self.run()  
  
 def draw(self):  
 self.screen.fill((37, 31, 33))  
 for model in self.objects:  
 model.draw()  
  
 self.bar\_button\_handler.update()  
 self.move\_buttons()  
  
 def get\_moving\_objects(self):  
 moving\_objects = []  
 if self.ready:  
 for index, model in enumerate(self.objects):  
 if type(model) is Object3D:  
 if model.is\_moving():  
 moving\_objects.append(index)  
 return moving\_objects  
  
 def move\_buttons(self):  
 for index, button in enumerate(self.all\_buttons):  
 rel\_pos = self.calc\_rel(index, button)  
 if rel\_pos:  
 button\_type = button.button\_type  
  
 if button\_type == 'model':  
 button.rel\_update\_pos(rel\_pos)  
 elif button\_type.startswith('axis'):  
 self.move\_axes(rel\_pos, button\_type)  
 elif 'vertex' in button\_type: # example: vertex2, where 2 is the object number  
 self.move\_vertices(rel\_pos, button\_type)  
 if not self.active\_button:  
 self.object\_vertex\_alter = None  
  
 def calc\_rel(self, index, button):  
 if button.on and self.active\_button is None: # if the user pressed on the button  
 self.active\_button = index  
 pg.mouse.get\_rel() # then we call this function for pygame to remember the mouse current position  
  
 elif index == self.active\_button: # if it's the same button that was pressed before  
 if button.on:  
 rel\_pos = pg.mouse.get\_rel()  
 return rel\_pos  
 else:  
 self.active\_button = None  
 return None  
  
 def move\_vertices(self, rel\_pos, button\_type):  
 rel\_pos = rel\_pos[0] / 120, -rel\_pos[1] / 126  
 v\_pos, x\_pos = button\_type.index('v'), button\_type.index('x')  
  
 object\_number, vertex\_number = int(button\_type[:v\_pos]), int(button\_type[x\_pos + 1:])  
 model = self.objects[object\_number]  
 model.move\_vertex(vertex\_number, rel\_pos)  
 model.axes.translate(model.get\_offset(model.calc\_center(), model.axes.calc\_center()))  
 self.object\_vertex\_alter = [object\_number, vertex\_number, model.vertices[vertex\_number]]  
  
 def move\_axes(self, rel\_pos, button\_type):  
 rel\_pos = rel\_pos[0] / 120, -rel\_pos[1] / 126  
 axis\_type = int(button\_type[4])  
 object\_number = int(button\_type[5:])  
 axis = self.objects[object\_number].axes  
 axis.vertices = self.moving\_axes\_custom(axis\_type, axis.vertices, rel\_pos)  
  
 @staticmethod  
 def moving\_axes\_custom(index, vertices, rel\_pos):  
 for vertex in vertices:  
 if index == 1:  
 dist = rel\_pos[1]  
 else:  
 dist = rel\_pos[0]  
 vertex[index] += dist  
 return vertices  
  
 def run(self):  
 while True:  
 if self.ready:  
 [(pg.quit(), exit()) for i in pg.event.get() if i.type == pg.QUIT or (i.type == pg.KEYDOWN and i.key == pg.K\_ESCAPE)]  
 self.draw()  
 self.camera.update()  
 pg.display.flip()  
 fps = str(int(self.clock.get\_fps()))  
 position = tuple(self.camera.pos)[:-1]  
 position = [math.floor(value \* 1000) / 1000 for value in position]  
 pg.display.set\_caption(f'FPS: {fps} | POS: {position}')  
 self.clock.tick(self.FPS)

Shapes:

import pygame as pg  
from Client\_Server.All\_3D.PreObject3D import PreObject3D  
from Client\_Server.All\_3D.Button import Button  
from Client\_Server.All\_3D.MatrixFunctions import \*  
import numpy as np  
OBJECTS\_FOLDER = r'Client\_Server/Project1/'  
  
# shapes' coordinates (homogenous)  
CUBE\_VERTICES = [(0, 0, 0, 1), (0, 1, 0, 1), (1, 1, 0, 1), (1, 0, 0, 1), (0, 0, 1, 1), (0, 1, 1, 1), (1, 1, 1, 1), (1, 0, 1, 1)]  
CUBE\_FACES = [(0, 1, 2, 3), (4, 5, 6, 7), (0, 4, 5, 1), (2, 3, 7, 6), (1, 2, 6, 5), (0, 3, 7, 4)]  
PYRAMID\_VERTICES = [(1, 0, 1, 1), (1, 0, -1, 1), (-1, 0, -1, 1), (-1, 0, 1, 1), (0, 1.6, 0, 1)]  
PYRAMID\_FACES = [[0, 1, 2, 3], [0, 3, 4], [0, 1, 4], [1, 2, 4], [2, 3, 4]]  
POINT\_VERTICES = [(0, 0, 0, 1)]  
POINT\_LINES = []  
LINE\_VERTICES = [(1, 0, 0, 1), (-1, 0.001, 0, 1)]  
LINE\_LINES = [(0, 1)]  
AXES\_VERTICES = [(0, 0, 0, 1), (1, 0, 0, 1), (0, 1, 0, 1), (0, 0, 1, 1)]  
AXES\_FACES = [(0, 1), (0, 2), (0, 3)]  
  
  
class Object3D(PreObject3D):  
 def \_\_init\_\_(self, renderer, vertices=None, faces=None, filename=None, pos=None, scaler=1.0, number=0):  
 axes = Axes(renderer, number)  
 axes.label = None  
 axes.scale(0.5)  
 if pos:  
 axes.translate(pos)  
 super().\_\_init\_\_(renderer, vertices, faces, filename, axes, number)  
 super().scale(scaler)  
  
 def draw(self):  
 super(Object3D, self).draw()  
 self.axes.draw()  
 self.translate(self.get\_offset(self.axes.calc\_center(), self.calc\_center()))  
  
 def scale(self, times):  
 self.vertices = self.vertices @ scale(times)  
 self.axes.scale(times)  
  
 def get\_all\_buttons(self):  
 return self.axes.buttons + self.object\_vertices  
  
  
class Cube(Object3D):  
 def \_\_init\_\_(self, renderer):  
 super().\_\_init\_\_(renderer)  
  
  
class Pyramid(Object3D):  
 def \_\_init\_\_(self, renderer):  
 super().\_\_init\_\_(renderer, np.array(PYRAMID\_VERTICES), np.array(PYRAMID\_FACES, dtype="object"))  
  
  
class Axes(PreObject3D):  
 def \_\_init\_\_(self, renderer, number=0):  
 super().\_\_init\_\_(renderer, np.array(AXES\_VERTICES), np.array(AXES\_FACES))  
 self.colors = [pg.Color('red'), pg.Color('green'), pg.Color('blue')]  
 self.color\_faces = [(color, face) for color, face in zip(self.colors, self.faces)]  
 self.label = 'XYZ'  
 self.draw\_vertices = False  
  
 self.buttons = [Button(self.screen, (0, 0), (0, 0), drag=True, buttontype=f'axis{i}{number}') for i in range(3)]  
  
 def draw(self):  
 super().draw()  
 self.update\_buttons()  
 for index, button in enumerate(self.buttons):  
 button.draw(True)  
  
 def update\_buttons(self):  
 for index, button in enumerate(self.buttons):  
 width\_height, center = self.calc\_button\_rect(self.object\_vertices[0].get\_pos(), self.object\_vertices[index + 1].get\_pos())  
 button.rect.width, button.rect.height = width\_height  
 button.rect.center = center  
  
 @staticmethod  
 def moving\_buttons\_func(index, vertices, rel\_pos):  
 index = -index-1  
 for vertex in vertices:  
 if index == 1:  
 dist = -rel\_pos[1]  
 else:  
 dist = rel\_pos[0]  
 vertex[index] += dist  
 return vertices  
  
 @staticmethod  
 def calc\_button\_rect(pos1, pos2):  
 width\_height = abs(pos1[0] - pos2[0]) + 10, abs(pos1[1] - pos2[1]) + 10  
 center = (pos1[0] + pos2[0]) / 2, (pos1[1] + pos2[1]) / 2  
 return width\_height, center  
  
 def calc\_center(self):  
 return np.array(self.vertices[0])  
  
  
class Line(PreObject3D):  
 def \_\_init\_\_(self, renderer):  
 super().\_\_init\_\_(renderer, np.array(LINE\_VERTICES), np.array(LINE\_LINES))  
  
  
class Point(PreObject3D):  
 def \_\_init\_\_(self, renderer):  
 super().\_\_init\_\_(renderer, np.array(POINT\_VERTICES), np.array(POINT\_LINES))  
  
  
class Circle(PreObject3D):  
 def \_\_init\_\_(self, renderer, vertices\_number=36):  
 vertices, faces = self.calc\_vertices(vertices\_number)  
 super().\_\_init\_\_(renderer, np.array(vertices), np.array(faces))  
  
 @staticmethod  
 def calc\_vertices(n): # n = number of vertices  
 angle = 2 \* math.pi / n  
 vertices = [(math.cos(angle \* i), math.sin(angle \* i), 0, 1) for i in range(n)]  
 faces = [[i for i in range(n)]]  
 return vertices, faces  
  
  
class Deer(Object3D):  
 def \_\_init\_\_(self, renderer):  
 super().\_\_init\_\_(renderer, filename=OBJECTS\_FOLDER + 'Assets/deer.obj', scaler=0.01)  
  
  
class Tank(Object3D):  
 def \_\_init\_\_(self, renderer):  
 super().\_\_init\_\_(renderer, filename=OBJECTS\_FOLDER + 'Assets/tank.obj', scaler=0.1)  
  
  
class Sphere(Object3D):  
 def \_\_init\_\_(self, renderer):  
 super().\_\_init\_\_(renderer, filename=OBJECTS\_FOLDER + 'Assets/sphere.obj')