**פרויקט גמר**

**מגיש: טל רווה**

הקבצים העיקריים המרכיבים את התוכנית:

קבצי מידע:

* elephant.fbx
* materials.txt
* help.txt

הקובץ elephant.fbx מכיל את כל המידע עבור הגאומטריה של המודלים (קואורדינטות של צמתים

ונורמלים עם אינדקסים אליהם) – מדובר במודלים אשר יצרתי בעצמי בתוכנת blender. כמו כן,

קובץ זה מכיל אובייקטים ריקים המייצגים את המיקומים של נקודות המבט והייחוס של המצלמות, את

נקודות ה- pivot של החלקים הנעים המרכיבים את הפיל ואת מיקום האור הנקודתי.

לכל אחד מהאובייקטים הנ"ל יש שם אשר משמש כמזהה בקוד (למשל, כל חלקי הפיל מתחילים עם

“e\_”).

הקובץ materials.txt מכיל את כל מידע עבור החומרים של המודלים (תחילה מופיע השם של המודל

ולאחר מכן הערכים עבור ה- diffuse, specular ו- shininess).

הקובץ help.txt מכיל את המידע הנדרש עבור המשתמש (בקרת מקלדת ועכבר).

קבצי קוד מקור:

* Header.h
* display.cpp
* input.cpp
* main.cpp
* menu.cpp
* readFunc.cpp
* transformations.cpp
* utilFunc.cpp

בקובץ Header.h מופיעים כל הקבועים, הצהרות על משתנים גלובליים, טיפוסים, שיטות וכד'.

בכל אחד מהקבצים האחרים מוגדרים המשתנים הגלובליים והשיטות הרלוונטיות לאותו קובץ.

הקובץ display.cpp מטפל בפעולות הקשורות לתצוגה (שיטת ה- callback של ה- display, שיטות

יצירת טקסטורה, וכד').

הקובץ input.cpp מטפל בפעולות הקשורות לקבלת קלט מהמשתמש (בקרת מקלדת, עכבר, מצלמה,

וכד').

הקובץ main.cpp מכיל את שיטת ה- main בלבד, אשר מבצעת את כל האתחולים הדרושים לפני

הפעלת התצוגה (אתחולים של glut, פונקציות ה- callback, תפריטים, תאורה, וכד')

הקובץ menu.cpp מכיל את השיטה אשר קובעת את התפריטים.

הקובץ readFunc.cpp מכיל שיטות אשר מטפלות בקריאת הקבצים הנוספים (elephant.fbx,

materials.txt, help.txt).

הקובץ transformations.cpp מטפל בכל הטרנספורמציות המבוצעות על המודלים הקיימים בסצנה.

הקובץ utilFunc.cpp מכיל שיטות עזר המשמשות לאורך כל התוכנית (חישובים של וקטורים בעיקר).

בנוסף לכל הקבצים הנ"ל יש (כמו בכל תוכנית שהגשתי) תיקייה המכילה את הקבצים של הספרייה

freeglut אשר יש לשימה בקונן C: (ולשנות את ה- solution platforms ל- x86) אם רוצים לקמפל את

התוכנית. קובץ ה- freeglut.dll מופיע כבר בתיקיית הפרויקט (Elephant World). כמו כן, שמתי עותק

שלו, ושל כל קבצי המידע, בתיקיית ה- Debug לשם הנוחות (שם גם נמצא קובץ ה- .exe, כמובן).

הסבר כללי לגבי אופן פעולת התוכנית:

תחילה קוראים את הקבצים elephant.fbx ו- materials.txt עבור המודלים וממפים את השמות שלהם

כדי שיהיה אפשר להתייחס אליהם לפי שם במהלך התוכנית. לאחר מכן מבצעים אתחולים בשיטת ה-

main של התוכנית.

השתמשתי בהחלשה (attenuation) ריבועית, עבור האור הנקודתי, ובהצללה חלקה לקבלת מראה טבעי

יותר של המודלים.

יש בתוכנית מספר אופציות עבור בקרת מצלמה ראשית (cam0):

* orbit
* pivot
* pan
* zoom

בשיטת orbit המצלמה מסתובבת סביב נקודת הייחוס שלה, ובשיטת pivot היא מסתובבת סביב עצמה.

את הפעולות הנ"ל בצעתי ע"י שימוש ב- spherical coordinates (ראה עמוד 398 בספר) כאשר מחליפים

בין ציר ה- y לציר ה- z.

בשיטת pan המצלמה זזה ימינה\שמאלה\למטה\למעלה – מבצעים זאת ע"י הוקטור למעלה (v) עבור

הזזה כלפי מעלה\מטה, וחישוב וקטור הצד (u) עבור הזזה ימינה\שמאלה, בהתאם להפרש קואורדינטות

מסך (נוכחיים מול קודמים).

בשיטת zoom פשוט מזיזים את נקודת המבט בכיוון של המצלמה (או בכיוון ההפוך עבור zoom out).

את המצלמה המייצגת את מבט הפיל (cam1) הגבלתי בהתאם לתזוזת הפיל (המראה ישתנה עם תזוזת

הגוף וגם עם תזוזת הראש). את תזוזת הראש והזנב החלטתי להגביל באופן סביר (למרות שעדיין אפשר

לעקם לו את הראש אם מנסים 😊).

את הטרנספורמציות עבור המודלים שומרים במטריצות מתאימות אשר מתעדכנות בכל פעם שיש תזוזה.

את תנועת הפיל מבצעים באמצעות שיטות הטרנספורמציה של openGL ומעדכנים את התכונות של

המצלמה cam1 בהתאם (כדי שיהיה תיאום בין המיקום והאוריינטציה של הפיל עבור שתי המצלמות).

יש שלוש שיטות היוצרות טקסטורה (אחת יוצרת לוח שחמט אדום-שחור והאחרות משתמשות בפונקציות

טריגונומטריות ליצירת גלים\עיגולים מעניינים – הגעתי לנוסחאות אלו באמצעות ניסוי וטעיה).

הטקסטורות מוצגות על גבי שלושת התמונות המופיעות בחנות.

את הרגליים של הפיל בחרתי דווקא להזיז באופן אוטומטי תוך כדי תזוזת הפיל כדי שהפיל יראה כאילו

הוא הולך (פשוט מחליפים כל פעם את הסימן של זווית הסיבוב ברגע שמגיעים לטווח התנועה המלא).

לצורך תצוגת המודלים משתמשים בטבלאות האינדקסים אשר נקראו מקובץ ה- .fbx (ניתן לקרוא על

מבנה קובץ כזה באתר <https://banexdevblog.wordpress.com/2014/06/23/a-quick-tutorial-about-the-fbx-ascii-format/>).

בפונקציית ה- display בודקים כל פעם אם יש תזוזה – אם כן אז מעדכנים את המטריצות המתאימות.

לאחר מכן עוברים ל- view (ע"י gluLookAt()), מעדכנים את מיקום האור ורצים בלולאה ומפעילים את

המטריצות ופונקציות הציור בהתאם למודל הנוכחי.

הסברים נוספים ומפורטים יותר מופיעים בתיעוד של קבצי המקור (ובקובץ help.txt).

כמה הערות חשובות:

1. הסצנה שבניתי מורכבת מלא מעט גאומטריה (קרוב ל- 8000 צמתים) ולכן כדאי להריץ אותה על

מחשב המצויד בכרטיס מסך (אחרת היא עלולה לרוץ לאט). בכל מקרה, אצלי היא עובדת מצוין

וצרפתי קובץ וידאו המדגים את אופן הפעולה המצופה של התוכנית.

2. לאחר הפעלת התוכנית (או לאחר הצגת חלון אחר) יש ללחוץ על החלון\המסך של התוכנית עם

העכבר פעם אחת כדי שהוא יהיה פעיל ויגיב כמו שצריך לקלט מהמשתמש.

3. את התפריט מפעילים עם הלחצן הימני של העכבר.