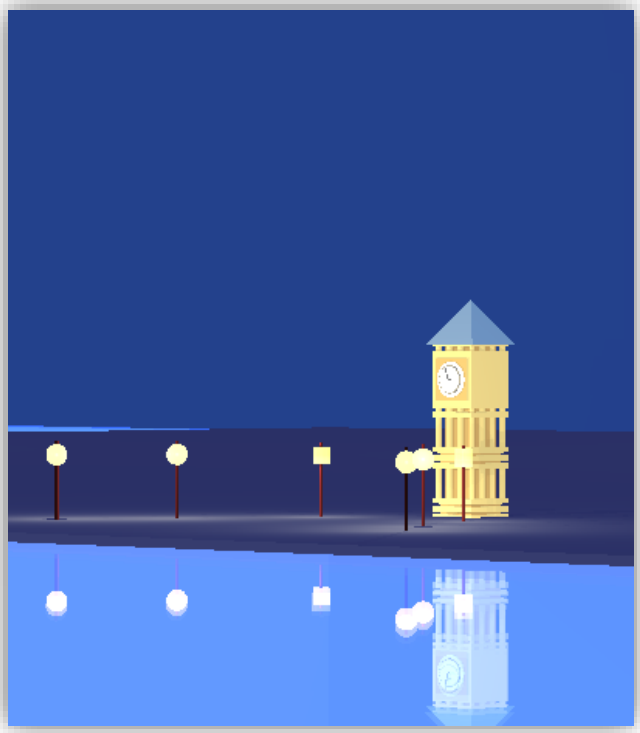
דו"ח מיני פרויקט מבוא להנדסת תוכנה

מגישות: מוריה מזרחי 212319692

יעל מקסימוב 212713069



יצירת תמונה

- ✓ בניין
- ✓ רקע(מנורות,רצפה,מים)
- ✓ שמים

יצירת בניין:

לשם יצירת הבניין יצרנו את המחלקות:

- Cube
- Base
- CylinderBase
- Clock
- Pyramid
- Building

Cube

- ❖ שדות המחלקה-6 פוליגונים כנגד 6 פאות של קובייה,8 נקודות כל 4 נקודות יצרו פוליגון שהוא כנגד פאה מסוימת בקובייה
- ❖ בנאי-מקבל 3 פרמטרים:נקודת התחלה ממנה נבנה את הקובייה,משתנה מסוג double עבור xz כלומר האורך והרוחב, ומשתנה double עבור הגובה-בהתאם לכך נבנת הקובייה.
- ❖ פונקציית getGeometries- שמחזירה את כל הפוליגונים-הריבועים שיצרנו
- ❖ פונקציית setCubeEmission- שמקבלת את צבע הקובייה-צבע אחיד לכל הפאות.
- ❖ פונקציית setCubeMaterial-שמאתחלת את החומר של הקובייה-כל הפאות מקבלות את אותו ערך עבור החומר.
- ❖ פונקציית getCubeTopNormal-שמחיזרה את הנורמל של הפוליגון העליון כך שהנורמל מכוון כלפי מעלה ולא כלפי מטה.(ישמש את המחלקות האחרות שישתמשו בקובייה.

Base-בסיס של הבניין 3 קוביות נמוכות הקובייה האמצעית קטנה מהאחרות

- ❖ שדות המחלקה-3 קוביות
- ❖ בנאי-מקבל 4 פרמטרים: נקודת התחלה ממנה נבנה את הבסיס,משתנה מסוג double עבור xz כלומר האורך והרוחב, משתנה double עבור הגובה-בהתאם לכך נבנת הקובייה, ומשתנה שקובע באיזה יחס הקובייה העמצאית תיהיה קטנה מהאחרות.
- ❖ פונקציית getGeometries- שמחזירה את כל הקוביות ע"י שימוש בפונקציית ה getGeometries שלהן.
- ❖ פונקציית setBaseEmission- שמקבלת 2 צבעים צבע אחד עבור הקובייה האמצעית וצבע שני עבור 2 הקוביות האחרות, כדי שנוכל ליצור מראה יותר תלת ממדי עבור הבסיס.
- ❖ פונקציית setBaseMaterial-שמאתחלת את אותו החומר עבור כל הקוביות

- פונקציית getC1Normal-שמחיזרה את הנורמל ממחלקת הקוביה עבור המחלקה שתירש ממנה CylinderBase
- ❖ פונקציית points-שמחזירה רשימה של הנקודות של הקובייה התחתונה לשימוש של המחלקה היורשת CylinderBase.

CylinderBase -רכיב בבניין שיורש מהבסיס ומוסיף עליו עמודי תווך-צילינדרים

- ❖ שדות המחלקה-16 צילינדרים
- ❖ בנאי- מקבל 4 פרמטרים: נקודת התחלה ממנה נבנה את הבסיס,משתנה מסוג double עבור xz כלומר האורך והרוחב, משתנה double עבור הגובה- בהתאם לכך נבנת הקובייה, ומשתנה שקובע באיזה יחס הקובייה העמצאית תיהיה קטנה מהאחרות.כל אלו ישלחו לבנאי האב, ובנוסף מתבצע חישוב להצבת הצילינדרים, בתחילה ישנה הצבה של הצילינדרים בכל אחת מ4 הפינות, ולאחר מכן יש חישוב של מרחק בין 2 צילינדרים וחלוקה ב3 כדי שנוכל להציב בין כל 2 עמודים עוד 2 עמודים במרחב שווה.
- ❖ פונקציית getGeometries- שמחזירה את הבסיס והצילינדרים.
- ❖ פונקציית setCylinderBaseEmission**-**שמקבלת 2 פרמטרים של צבע שנשלחים לבסיס, וקובעת שצבע הצילינדרים יהיה כצבע הקוביה הראשונה והשלישית בבסיס.
- ❖ פונקציית setCylinderBaseMaterial שקובעת את אותו החומר עבור כל הצילינדרים.

Clock

- ❖ שדות-פוליגון-כנגד הריבוע שמאחרוי השעון,3 צילינדרים-שיהיו שטוחים כדי ליצור את המעגלים של השעון, 2 פוליגונים מלבנים- כנגד המחוגים
- ❖ בנאי שמקבל 2 פרמטרים- נקודת התחלה ופרמטר שיקבע את האורך והרוחב של הריבוע שימסגר את השעון,בבנאי מתבצעים חישובים של גודל השעון ביחס לריבוע שניתן,הזזות מתאימות של הצילינדרים כדי שיהיו אחד על גבי השני, וכן חישובים של מיקום המחוגים ביחס למרכז ולעיגול העליון.
- ❖ פונקציית setClockEmission-שמקבלת 2 פרמטרים צבע עבור העיגולים של השעון ושל הריבוע הממסגר את השעון, צבע העיגול האמצעי(שיוצר סוג של עיגול שחור דק על השעון) והמחוגים נקבע כשחור.
- ❖ פונקציית setClockMaterial-שקובעת את החומר של העיגולים והריבוע.

Pyramid

- ❖ שדות-4 משולשים ופוליגון
- ❖ בנאי- מקבל 3 פרמטרים:נקודת התחלה ממנה נבנה את הפירמידה,משתנה מסוג double עבור xz כלומר האורך והרוחב, ומשתנה double עבור הגובה- בהתאם לכך נבנת את הפירמידה בבנאי מתבצע חישוב של המרכז של הריבוע שממנו אנו מעלים את הגובה של הפרמידה ובעזרת 5 הנקודות שחושבו בבנאי אנו בונים את המשולשים והריבוע.
- ❖ פונקציית getGeometries-שמחזירה את את כל השדות של המחלקה.
- ❖ פונקציית setPyramidEmission-שקובעת את הצבע

- פונקציית setPyramidMaterial-שמאתחלת את החומר

Building

- שדות-קבועים עבור צבעים וחומרים,ורכיבי הבניין
- בנאי-שמקבל 2 פרמטרים נקודות התחלה ומשתנה אורך ורחוב שבעזרתם בנינו את הבניין
- כך ש:

1. כנגד החלק התחתון של הבניין יצרנו 2 בסיסים אחד מעל השני כל אחד בגובה 4.5
2. מעליהם ישנו בסיס עם צילינדרים בגובה 20
3. מעליו עוד 2 בסיסים כל אחד בגובה 4.5
4. מעליהם בסיס עם צילינדרים בגובה 20
5. מעליו בסיס צילינדרים נמוך יותר בגובה 5
6. מעל קוביה בגובה 30 שעליה יצרנו את השעון והזזנו אותו קדימה כדי שיהיה ניתן לראות אותו
7. עוד בסיס צילינדרים נמוך...
8. ולבסוף פירמידה בגובה 25

- פונקציית getGeometries-שמחזירה את כל השדות

רקע

עבור הרקע יצרנו:

מים- 2 מישורים מקבילים במרחק קצר אחד מהשני כך שהמשטח התחתון כחול והעליון שקוף ומשתקף.

קרקע-יצרנו קוביה נמוכה רחבה וארוכה

הוספנו 6 מנורות רחוב:

4 מנורות מסוג StreetLamp שהתאורה שלהם כלואה בכדור שקוף-תאורה מסוג פוינט לייט.

2 מנורות מסוג SpotStreetLight שהתאורה שלהם כלואה בצילינדר שקוף-התאורה מסוג ספוט לייט.

וכן הוספנו אור כיווני כדי לכלול את כל התאורות.

שמים

עבור שמים יצרנו 3 מישורים שיהיו מאונכים לרצפה ויהיה מרחב בינהם כמו בחדר

ועל כל משטח פיזרנו כוכבים במרחק קצר מהמשטח

ועבור המשטח שנראה בתמונת המול הצבנו ירח שזה 2 צילינדרים אחד לבן אחד בצבע השמים מעליו כך שהצל שלו יוצר סהר.

:MP:1

antialiasing

נידרשנו לעשות antialiasing בשלב 8, בעקבות באג של קצוות משוננים שהתגלה לנו בתמונה.

כדי לתקן את באג זה הוספו 2 פונקציות במחלקת Camera, שישלחו כמה קרניים לכל פיקסל בניגוד למה שהיה לנו עד כה(נשלח רק קרן אחד למרכז הפיקסל), נסכום את הצבעים המוחזרים של הקרניים ונחלק במספר הקרניים, דבר זה יצור לנו את ממוצע הצבעים של הפיקסל.

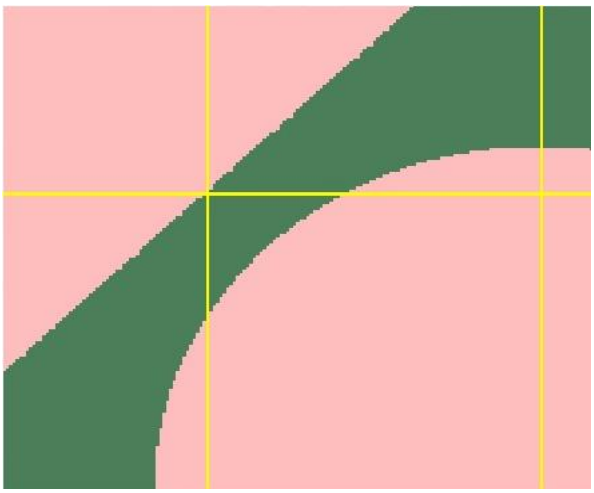
כדי לבצע את שיפור זה נשתמש באלגוריתם Super-sampling.

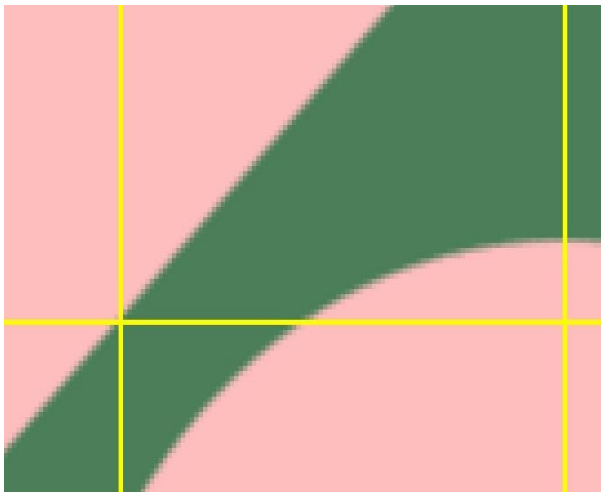
פונקציה אחת מקבלת את אורך ורוחב התמונה ומיקום הפיקסל ע"י i,j ויוצרת בעזרתם קרניים רנדומליים בתוך הפיקסל. להלן פעולות של הפונקצייה:

- יצירת רשימה של קרניים כדי לאחסן את הקרניים שניצור.
- נוסיף לרשימה את הקרן עבור מרכז הפיקסל הנוכחי על ידי קריאה לפונקציה `constructRay` עם הפרמטרים המתאימים.
- נחשב את המידות של כל פיקסל על ידי חלוקת הרוחב והגובה הכוללים של התמונה ב-'nX' ו-'nY'- אורך ורוחב התמונה, בהתאמה.
- נחשב את גורמי קנה המידה (`yScale` ו-`xScale`) עבור קואורדינטות x ו-y בהתבסס על מיקום הפיקסל הנוכחי ביחס למרכז התמונה.
- נתאים את נקודת ה-pixelCenter על ידי הוספת המרחקים המתאימים בכיווני vRight ו-vUp כדי להגיע לאמצע התמונה.
- ניצור אובייקט 'אקראי' כדי ליצור ערכים אקראיים עבור דגימת-על.
- נפתח ללואה ונכנס אליה nss פעמיים- כמות הקרניים אותו נרצה.
- עבור כל איטרציה, ניצור גורמים אקראיים ע"י הגרלה משתנה בוליאני עם המשתנה הינו אמת נגריל מספר חיובי ואם המשתנה הינו שקר נגריל מספר שלילי.
- נחשב את `dx ו-dy על ידי הכפלת הגורמים האקראיים במידות הפיקסלים.
- ניצור `RandomPoint` על ידי הוספת מכפלת dx ו dy ל vRight ו vUp בהתאמה לנקודת `pixelCenter` אם dx I dy לא אפס.
- ניצור קרן חדשה עם המקור הנקודה 0p וכיוון מ0p לנקודה רנודמלי ונוסיף לרשימה.
- נחזיר את רשימת הקרניים שיצרנו.

הפונקציה השניה מקבלת את רשימת הקרניים, עוברת על כל קרן וסוכמת את הצבעים שהתקבלו מכל קרן ע"י פונקציית tracTracer ומחלקת בnss (מספר הקרניים) משתנה שמקבל ערך ביצירת הטסטים.

דוגמא:(לפני->אחרי)





adaptive antialiasing:MP2

נידרשנו לעשות adaptive antialiasing בשלב 9, בעקבות זמן ריצה מטורף של השיפור משלב 8.

כדי לשפר את זמן הריצה הוספנו 2 פונקציות.

אחת תקבל i ו j למיקום הפיקסל, תיצור כמו שיצרנו עד עכשיו קרן מרכזית לפיקסל ע"י הפונקציה constructRay, תקבל את צבע הפיקסל ע"י הפונקציה rayTracer. הפונקציה תקרא לפונקציה השניה ותשלח אליה את אורך ורוחב התמונה, i l j משתנה maxLevel משתנה שמקבל את ערכיו ביצירת הטסט, בנוסף הפונקציה תשלח את צבע הקרן המרכזית של הפיקסל.

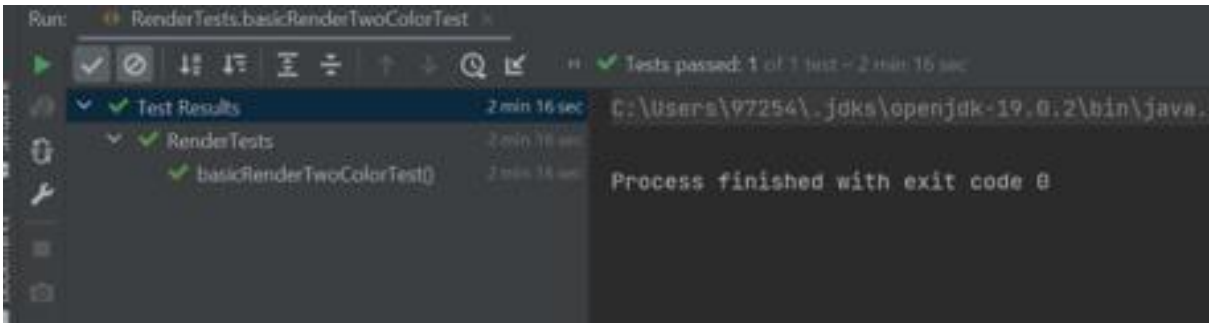
הפונקציה השניה תשלח את הקרנים ותחשב את הממוצע. הפונקציה תקבל את כל המתשנים הנל, תבדוק אם משנה של רמת המקסימום 0 אם כן תחזיר את הצבע של קרן מרכז הפיקסל אם לא, הפונקציה תיצור מערך, תכניס לתוכו 4 קרניים הממוקמות ב4 רביעי ציר xy בתוך הפיקסל הנוכחי כך שקרן המרכז תפגע במרכז הפיקסל. לאחר מכן הפונקציה תקבל את הצבעים של 4 הקרניים, תשווה עם צבע מרכז הפיקסל. אם הצבע תואם אין טעם להמשיך בשליחת קרניים לאזור בעלי צבע אחיד, אם לא נפצל-נבצע קריאה רקורסיבית לאותה פונקציה ונוריד ברמה את משתנה maxLevel ליציאה מהרקורסיה.

לאחר מכן נסכום את הצבעים שהתקבלו מאותו אזורים, נחלק ב 5 (קרן מרכז הפיקסל גם נכללת).

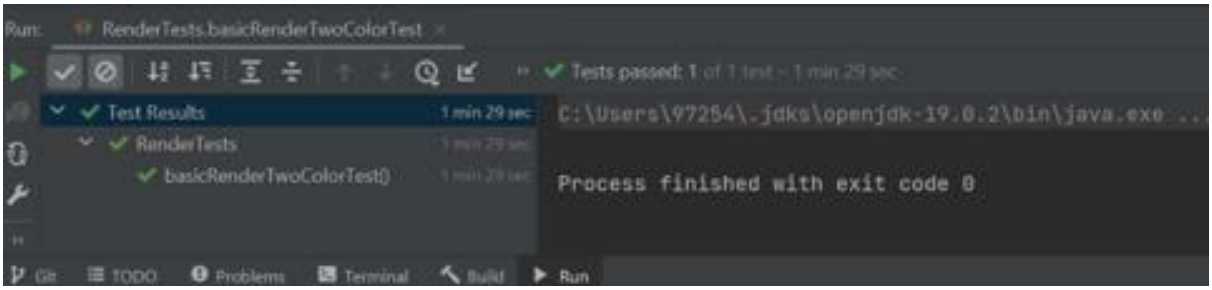
הרעיון הוא חיסכון חישוב של קרניים בעלי אותו צבע.

בתמונות למטה נוכל להשוות את זמני הריצה ולראות את השיפור.

לפני:



אחרי:



שיפור ב47 שניות.

BVH:MP2

שיפור זה נועד לקצר את זמני הריצה על ידי יצירת קופסא תוחמת לכל אחת מהצורות, כך שבהישלח קרן מהמצלמה לא נחשב ישר את החיתוכים של הקרן עם הגוף אלה תחילה נבדוק האם הקרן בכלל חותכת את הגוף על ידי בדיקה זולה יותר שנעשה על הקרן, שבה נבדוק האם הקרן בכלל עוברת דרך הקופסא שיצרנו עבור הגוף, ורק במקרה שהיא אכן עוברת בקופסא נחשב את החיתוכים.

✓ לשם כך במחלקת Intersectable:

יצרנו 2 שדות חדשים-

- שדה בוליאני שבודק האם השיפור דלוק או לא שיאותחל לערך ברירת מחדל true .
- קופסא תוחמת-עבור שדה זה פתחנו מחלקה פנימית בתוך Intersectable בשם AABB שיהיו בה 2 נקודות, נקודה אחת כנגד כל ערכי המקסימום של xyz ושניה כנגד כל ערכי המינימום.

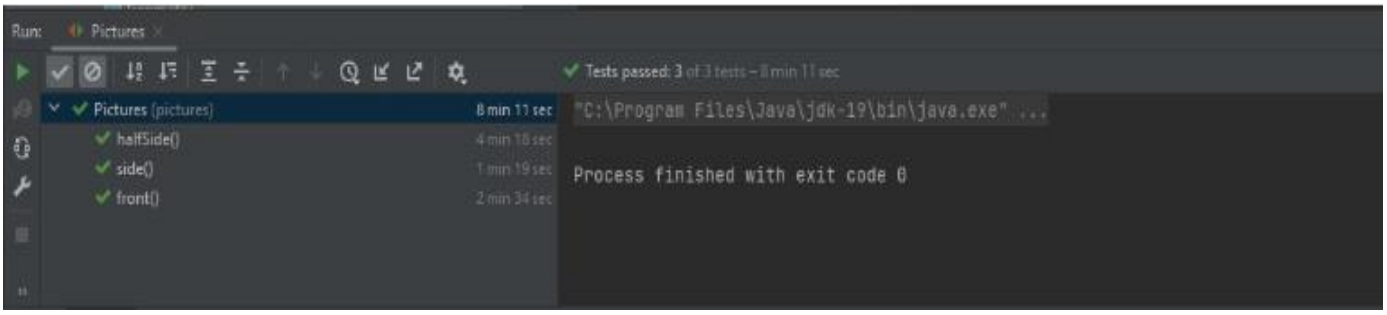
-בנוסף הוספנו למחלקה פונקציה שבודקת האם הקרן חותכת את הקופסא התוחמת ומחזירה true/false בהתאם.

- ✓ במחלקות הגופים:קראנו לפונקציה שיוצרת AABB בסוף הקונסטרקטור של הגוף.
- ✓ במחלקת Geometries :
  - הוספנו פונקציה שיוצרת קופסא עבור מאגד של גופים ע"י בדיקת הערכי xyz המקסימליים והמינימליים של כל הקופסאות.
  - בתחילת פונקציית הקונסטרקטור ובפונקציית aadd קראנו לפונקציה שבונה את הקופסא עבור קבוצת הגופים
  - בפונקציה
- ✓ במחלקת Intersectable:

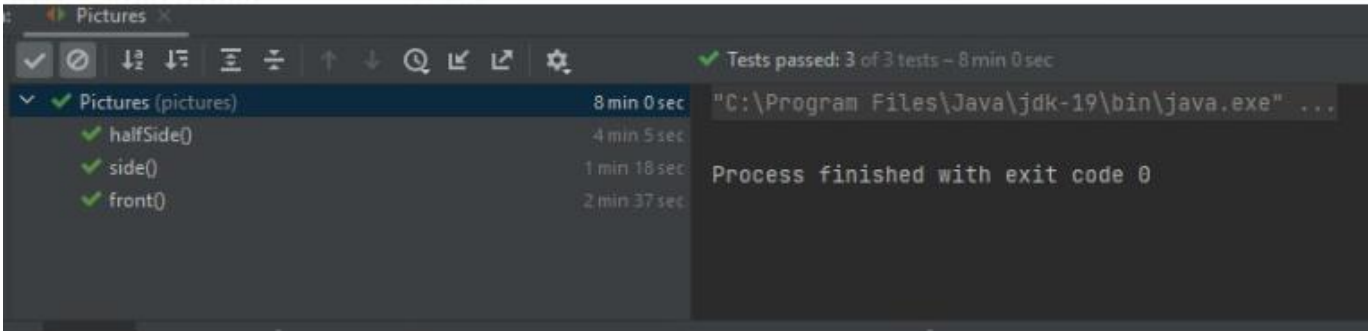
הוספנו לפונקציה findGeoIntersections תנאי שבודק האם השיפור מופעל ומפעיל את הפונקציה לבדיקת קרן,אם אחת מהתשובות לתנאי הם לא, יוחזר null עבור הקרן ולא נמשיך לחישוב החיתוכים.

תמונות של זמני ריצה:

בלי BVH:



עם BVH:



ניתן לראות שזמן הריצה שופר ב11 שניות.

בונוסים:

-צילינדר

-פוליגון

- פונקציית findGeoIntersectionsHelper עם maxDistance

קרדיטים:

-הילה בונזח- עבור עזרה בקוד הBVH ובשיפורים.

-המרצה אליעזר גנסבורגר-על הקוד שפורסם בגיט ועל הלימודים במהלך הסמסטר.

-chatGPTו -עבור התיעוד ועזרה בשאלות על השפה.

קישור לפרויקט-https://github.com/YaelMaximov/ISE5783_9693_3069

תודה רבה!! 😊