ב"ה

פרוייקט רובוט חשאי-פגישה #2-שירה עוזרי

מנחה: פרופ' רועי פורן

הרעיון הכללי:

נרצה ליצור רובוט שמנווט מנק' התחלה לנק' יעד תוך שמירה על חשאיות מירבית.
בעיות עם ההגדה: מה זה חשאיות?

הבעיות שכרגע אני מתעסקת בהן:

1) מיפוי-בהנתן אזור נתון, שנניח שהוא מצולם מכמה זויות, ייצור "תמונת מצב" טובה של האזור.

בשלב המיפוי, המטרה היא להבין כיצד נראה המרחב מכמה זוויות שונות – כאילו הוצבו בו מצלמות במקומות שונים. הרעיון הוא לזהות אילו אזורים בכיתה גלויים לעין (ולכן מסוכנים עבור הרובוט), ואילו מוסתרים (ולכן בטוחים). כדי לעשות זאת, נשתמש במצלמות וירטואליות בסביבת Unity, שיצפו על אובייקט ה־Classroom שהורדתי מזוויות שונות. בעזרתן נוכל לבנות "מפת חשיפה" – כלומר, להבין אילו אזורים רואים מכל מצלמה. בהמשך, מידע זה יהפוך לבסיס ל־Heatmap של רמות סיכון, שישמש את הרובוט לחישוב מסלול בטוח.

הפיתרון שלי כרגע-

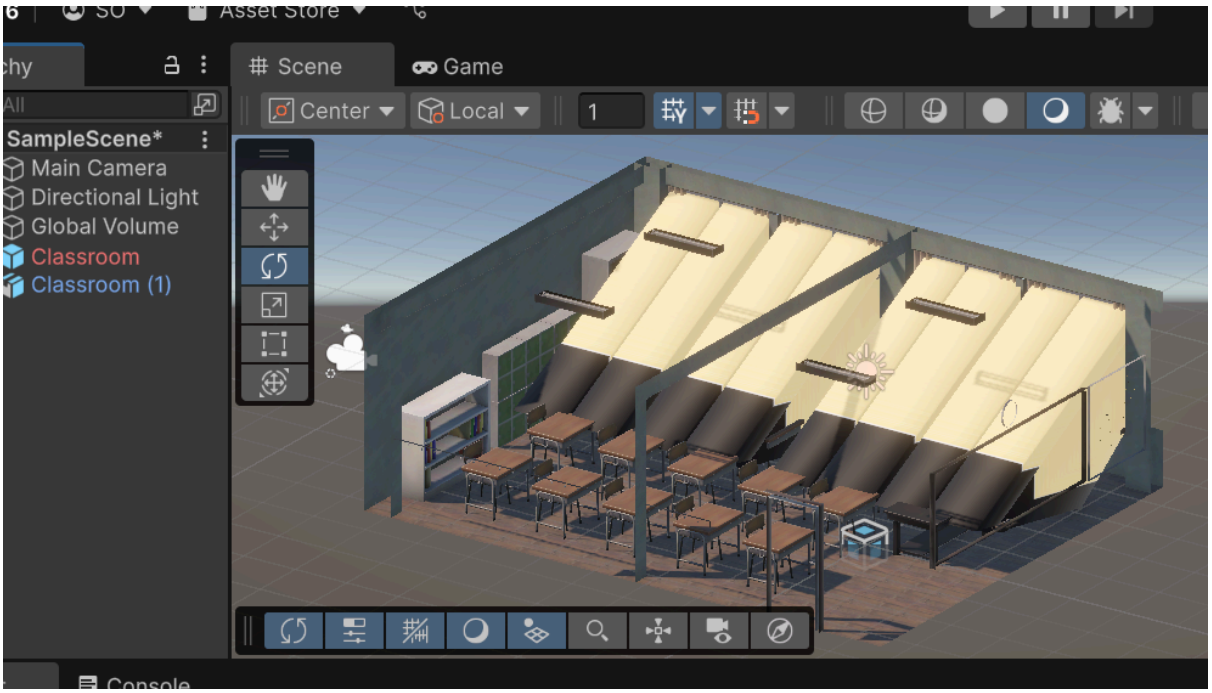
הפתרון שלי לשלב המיפוי הוא השימוש במודל התלת־ממדי של כיתה (Classroom) שהורדתי והטענתי ל־Unity. המודל הזה מאפשר לי לדמות בצורה ויזואלית ומרחבית את מבנה הסביבה שבה הרובוט יפעל, כולל פרטי ריהוט, קירות וגבולות תנועה. בכך, הוא מהווה תחליף לצילום מהעולם האמיתי מזוויות שונות – שכן אני יכולה למקם מצלמות וירטואליות בכל נקודה שארצה ולנתח אילו אזורים נחשפים לעיניהן. באמצעות המודל הזה, אני יוצרת "תמונת מצב" עשירה שממנה ניתן לגזור מפת חשיפה – שלב הכרחי לקראת בניית heatmap וחישוב מסלול רובוטי בטוח תוך שמירה על חשאיות.

מעבר מהעולם האמיתי והבעיה הפיזית שאני רוצה לפתור לסימולציה-

הסימולציה ב־Unity מאפשרת לי לגשר בין הבעיה הפיזית האמיתית של ניווט רובוטי חשאי לבין פתרון שניתן לחשב ולבחון. על ידי בניית סביבה וירטואלית שמדמה את הכיתה, וחלוקתה לגריד עם ערכי חשיפה, אני יוצרת ייצוג מתמטי של העולם. ייצוג זה מאפשר לי לבדוק אלגוריתמים, לבחון תרחישים ולבצע אופטימיזציה למסלול – בצורה בטוחה, מבוקרת וחוזרת, מבלי להזדקק לרובוט פיזי או לציוד יקר בשלב הזה.

לסיכום-

כרגע אני "אניח" שנבנה מודל של החלל בו הרובוט שלי פועל.(הנחה1)



https://sketchfab.com/3d-models/classroom-d4553cc2008242849214e4cbf8ad8551

2) מציאת אזורים "בטוחים"-הגדרה "אזור בטוח", יצירת heat map בהתאם למיפוי משלב 1.

הבעיה:

במערכת שלנו, אנחנו מנסים לפתור את בעיית **הערכת רמת החשיפה של רובוט בסביבה סגורה** (למשל בתוך כיתה). המטרה היא להעריך, לכל נקודה ברצפה, עד כמה היא "בטוחה" — כלומר, עד כמה הסיכוי שהרובוט יתגלה כשהוא עומד שם הוא נמוך. כדי לעשות זאת, אנחנו מחשבים לכל משבצת ברצפה ציון שמבוסס על שלושה גורמים מרכזיים: (1) עד כמה המשבצת חשופה לאור ישיר ממקורות תאורה או חלונות, (2) עד כמה המשבצת מוסתרת על ידי אובייקטים כמו שולחנות וכיסאות, ו-(3) עד כמה המשבצת קרובה לדלתות (אזורי כניסה בעייתיים). בעזרת שקלול של שלושת הגורמים האלה, אנחנו יוצרים Heatmap — מפה צבעונית שבה אזורים בטוחים מופיעים בירוק ואזורים מסוכנים באדום. בצורה כזו, ניתן להבין במהירות מהם האזורים האידיאליים לתנועה נסתרת של הרובוט.

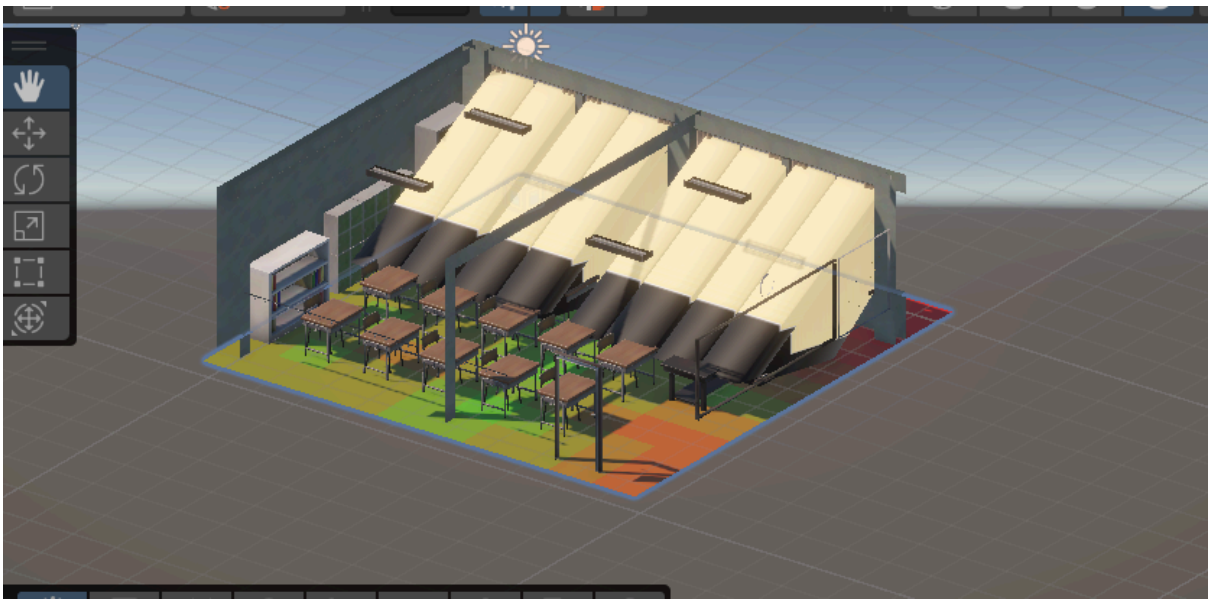
הפיתרון שלי:

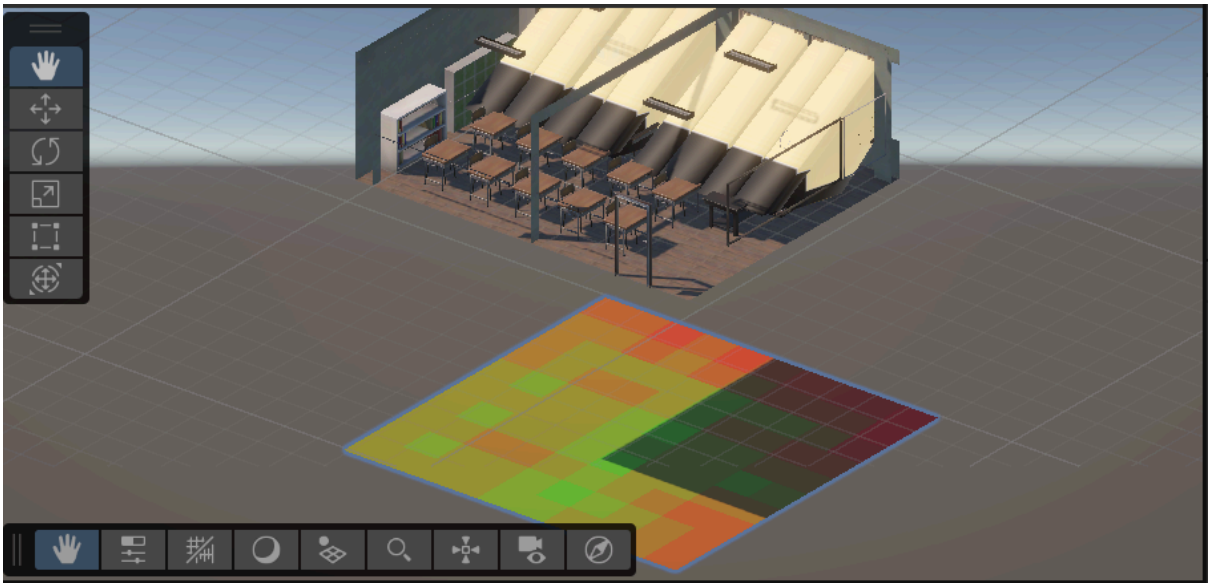
הפתרון שפיתחנו לבעיית הערכת החשיפה מבוסס על חישוב חכם של שלושה גורמים מרכזיים המשפיעים על מידת הבטיחות של הרובוט בכל נקודה בחדר: מידת החשיפה לאור ישיר, מידת ההסתרה על ידי אובייקטים פיזיים, ומידת הקרבה לאזורי סיכון כמו דלתות. עבור כל משבצת ברצפה, המערכת מחשבת שלושה ציונים נפרדים בטווח 0–100 המייצגים את עוצמת האור הישיר, את רמת ההסתרה, ואת מידת הקרבה לדלת, בהתאם להשפעתם על גילוי הרובוט. שלושת הציונים משוקללים יחד לפי משקלים שנבחרו בקפידה, כך שניתן משקל גבוה יותר לגורמים המרכזיים (אור והסתרה), ומשקל קטן יותר לקרבה לדלת, בהתאם לחשיבות היחסית שלהם בזיהוי בסביבה אמיתית. לאחר מכן, הציון הכולל מתורגם לצבע ב-Heatmap, כאשר ירוק מייצג אזור בטוח ואדום אזור מסוכן. הפתרון יעיל משום שהוא מחשב בצורה אוטומטית את מאפייני הסביבה מבלי להסתמך על קונפיגורציה ידנית, והוא הגיוני כי הוא משקלל את הגורמים המשפיעים בצורה ריאלית על זיהוי פיזי בעולם האמיתי. בכך, המערכת מאפשרת לרובוט לקבל החלטות תנועה חכמות ומהירות, ומייעלת משמעותית את היכולת שלו לשמור על חשאיות.

מבחינת הקוד:

```
for (int x = 0; x < rows; x++)
{
    for (int z = 0; z < cols; z++)
    {
        Vector3 position = new Vector3(x * cellSize, 0.01f, z * cellSize);
        Quaternion rotation = Quaternion.Euler(90, 0, 0);
        GameObject cell = Instantiate(cellPrefab, position, rotation, transform);

        // חישוב שלושת הפונקציות
        float f1 = ComputeLightExposure(position, lightSources);
        float f2 = ComputeObstacleCover(position, obstacles);
        float f3 = ComputeDoorDistance(position, doorPos);

        // שקלול סופי לפי משקלים
        float finalScore = (f1 * 0.49f) + (f2 * 0.49f) + (f3 * 0.02f);

        // קוונטיזציה ל-10 דרגות
        float quantizedScore = Mathf.Round(finalScore / 10f) * 10f;

        // צבע המשבצת לפי רמת החשיפה (0 = ירוק, 100 = אדום)
        Color heatColor = Color.Lerp(Color.green, Color.red, quantizedScore / 100f);

        Renderer renderer = cell.GetComponent<Renderer>();
        renderer.material = new Material(renderer.material);
        renderer.material.color = heatColor;
    }
}
```

לאחר הרצה:





הבעיה הבאה שיש להתמודד איתה:

חישוב מסלול בטוח-בהנתן הheatmap וoracle שפותר את בעית הaudio(שנמצא data עבורו), נחשב את המסלול האופטימלי בין נק' ההתחלה ליעד.