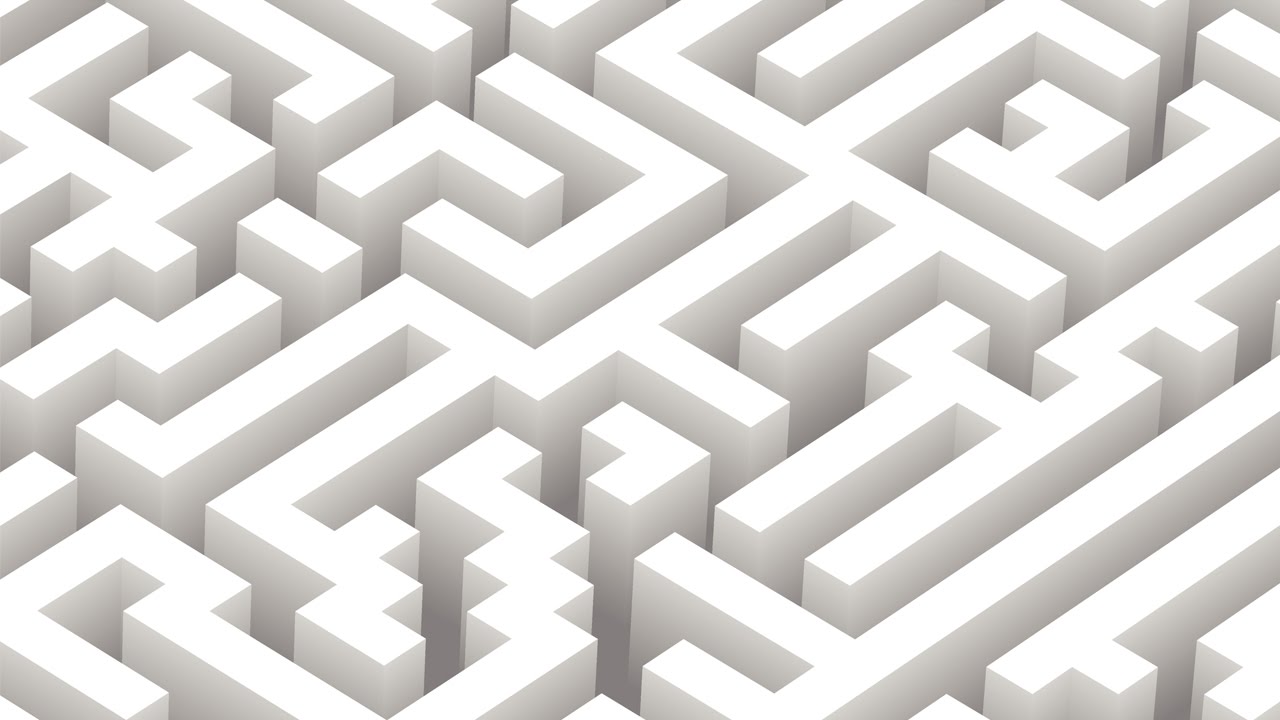


The Maze

פרוייקט 5 יח"ל - סייבר רשתות

בתכנון ותכנות מערכות סייבר



שם התלמיד : מתן

שם משפחה: עודד

ת"ז: 322281551

כיתה: י"ב 3

ביה"ס: תיכון אוסטרובסקי

מורה מלווה: אתי בררו

קיץ 2018

תוכן עניינים

[מבוא 3](#_Toc512800791)

[אודות המשחק 4](#_Toc512800792)

[מהלך המשחק 4](#_Toc512800793)

[אסטרטגיות 4](#_Toc512800794)

[טיפוסי נתונים במשחק 5](#_Toc512800795)

[קבצים 6](#_Toc512800796)

[תרשים זרימת המשחק 8](#_Toc512800797)

[מצב רגיל/מרובה משתתפים 8](#_Toc512800798)

[מצב משחק נגד השעון 9](#_Toc512800799)

[מבנה המסך 10](#_Toc512800800)

[מצב משחק נגד השעון 11](#_Toc512800801)

[פתרון המבוך 12](#_Toc512800802)

[מסך ניצחון/הפסד 13](#_Toc512800803)

[מסך אפשרויות מרובה-משתתפים 14](#_Toc512800804)

[תיאור המחלקות – פעולות מרכזיות 15](#_Toc512800805)

[אלגוריתם מרכזי 15](#_Toc512800806)

[תא במבוך 17](#_Toc512800807)

[יצירת התאים במבוך והגדרת השכנים שלהם 19](#_Toc512800808)

[טיפול בתזוזה 20](#_Toc512800809)

[חיבור בין 2 חוליות של המבוך 22](#_Toc512800810)

[השרת 23](#_Toc512800811)

[הלקוח 26](#_Toc512800812)

[קשיים 30](#_Toc512800813)

[מסקנות 31](#_Toc512800814)

[ביבליוגרפיה 32](#_Toc512800815)

# מבוא

בחרתי לעסוק בנושא זה כיוון שהוא משלב עבורי כלים רבים שרלוונטיים לעולם התכנות – Events, Threading, UI ועוד. כל אלו, באים לידי ביטוי ברמה גבוהה בשפת C# שבה השתמשתי כיוון שהכרתי כבר את השפה Java וידוע שהשפות דומות אחת לשנייה. רציתי ללמוד דברים נוספים ב-C# ועל כן החלטתי לפתח את הפרויקט בשפה זו.

לסביבת העבודה WPF נחשפתי במסגרת הסמכה של מיקרוסופט שעשיתי, ורציתי ללמוד יותר על התכונות השונות של הסביבה. על כן, החלטתי לפתח את משחק המבוך, שידעתי שאוכל לפתח בסביבה זו בעזרת השקעה והתמדה.

המבוך הוא רק דוגמה למגוון רחב של משחקים שאפשר לפתח, וחשבתי שהוא יהיה לי מצוין ללמידה רחבה של כל הכלים הדרושים. לא רציתי לקחת נושא שהיה חשש שלא אוכל לממש אותו כמו שצריך, והחלטתי לבחור נושא שמבחינה רעיונית אני יודע שאני מסוגל לפתח, וכן להציב לעצמי אתגרים שונים במהלך הפיתוח להעמקת הידע שלי בצורות שונות.

החלטתי להוסיף מערכת של משחק מרובה-משתתפים, כי רציתי ללמוד כיצד ניתן להעביר תקשורת בין כמה מחשבים. לא ידעתי הרבה על צורת הפיתוח של מערכת כמו זו, ועל כן לאחר קריאה רבה בנושא הצלחתי לפתח אותה בדרכי.

בתיק פרויקט זה אפרט את רכיבי הפרויקט שלי – באילו כלים השתמשתי, כיצד, הפונקציונליות של הקוד וכן ארד לפרטים של צורת עבודתו.

# https://upload.wikimedia.org/wikipedia/commons/c/c9/Picture_maze_unsolved.pngאודות המשחק

מבוך הוא חידה הבנויה ממעברים מתפצלים, אשר על הפותר למצוא נתיב דרכה. המבוך שונה מן **הלבירינת**, שבו מסלול אחד בלבד, והוא לא נועד להקשות על מציאת הדרך בו. המבוך עשוי להיות בנוי חדרים המובילים לחדרים אחרים, כפי שמצוי לעיתים קרובות במשחקי תפקידים בסגנון מבוכים ודרקונים: במובן זה, החדרים מתפקדים כמעברים.

בפרויקט זה, מימשתי צורה של **לבירינת**, שכן ישנה רק דרך אחת להגיע לסוף המבוך, ואין חדרים מתפצלים אלא דרכים אחרות שבסופן יש קירות ללא מוצא.

בפרוייקט זה המבוך הוא גם מרובה משתתפים, דבר שהופך את המשחק לאינטראקטיבי יותר ומהנה יותר, וכן מאפשר למשתמש להתחרות נגד משתמשים אחרים במטרה לנצח.

## מהלך המשחק

השחקן יתחיל את דרכו בפינה השמאלית העליונה של הלבירינת, שתמיד תהיה נקודת ההתחלה. השחקן ינסה למצוא את דרכו אל נקודת הסיום. כאשר יגיע אליה, יקבל הודעה שהצליח להגיע לנקודת הסיום ותינתן לו האפשרות לשחק שוב.

כמו כן, ישנו מצב מרובה המשתתפים, בו לפני שהמבוך נוצר, המשתמש נדרש להתחבר אל משחק, או ליצור משחק משל עצמו, ואז הוא תלוי במשתמשים האחרים באותו המשחק.

בנוסף לכך, ישנו מצב משחק נגד השעון, בו המשתמש צריך לסיים את המשחק לפני שהזמן מגיע ל-0, ובכך המשחק נהיה מאתגר יותר.

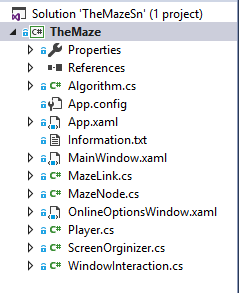
## אסטרטגיות

כמו במבוכים רגילים, השחקן יכול להסתכל בעיניו על נקודת הסיום, ומשם לתכנן את דרכו אל נקודת ההתחלה/נקודה בה הוא יודע שהוא יכול להגיע אליה מנקודת ההתחלה. משם הוא פשוט ימשיך על אותה הדרך עד אשר יגיע לסוף.

# טיפוסי נתונים במשחק

* מערך של 'MazeNode' אשר משמשים כתאים במבוך. לפי מערך זה ניתן לדעת את מיקום התאים במבוך.
* רשימה של חיבורים בין 2 'MazeNode' (MazeLink)
* Tuple בו השתמשתי כדי לשלוח את מיקומי החוליות במבוך, ומי קודמיהן.

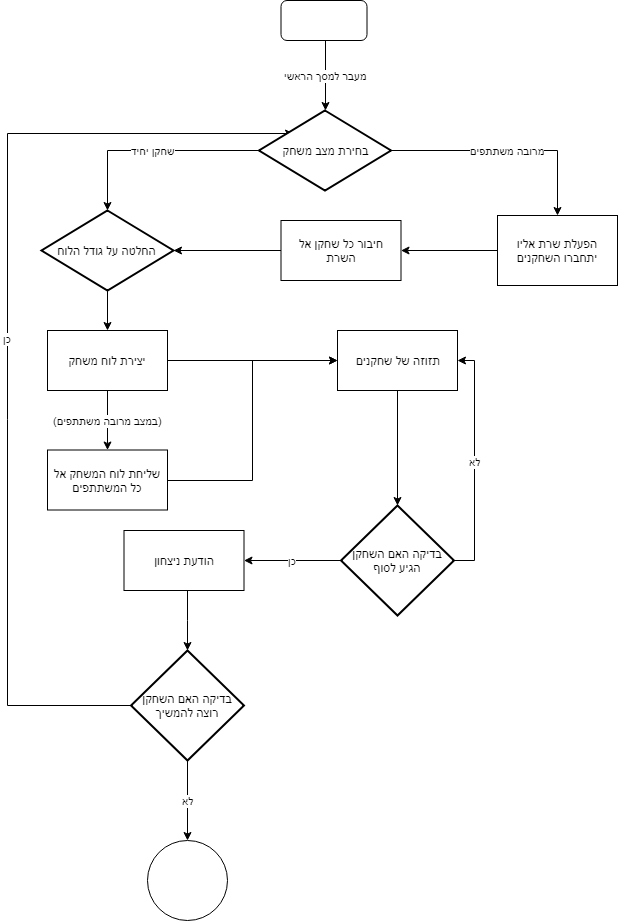
# קבצים



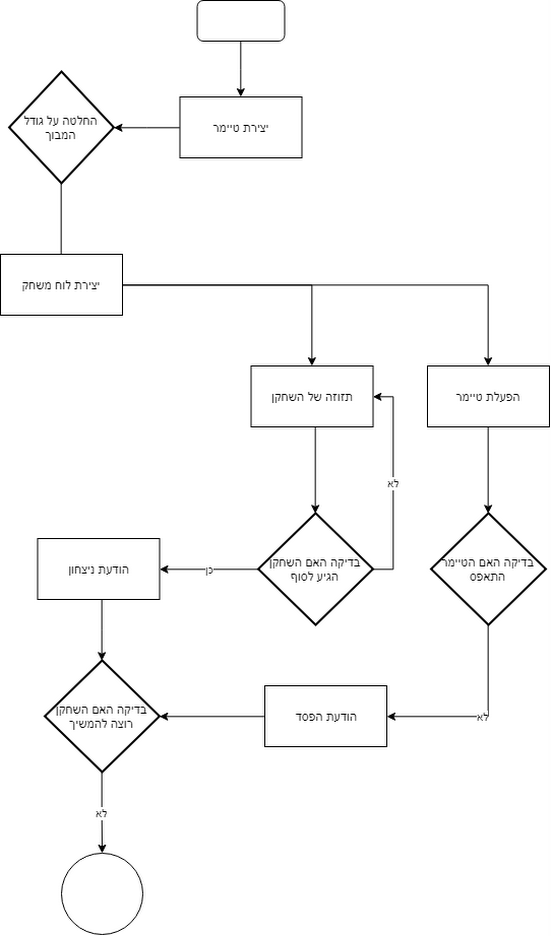
* **TheMaze** – הפרוייקט עצמו
* **Algorithm.cs** – מכיל פעולה אחת והיא FindSpanningTree שתפקידה היא לבנות את המבוך
* **MainWindow.xaml** – החלון הראשי, מכיל גם את החלק הלוגי **MainWindow.xaml.cs** בו כתובים החלקים שאחראיים לניהול מהלך המשחק, קריאה לפעולות המרכזיות וכו'
* **MazeLink.cs** – המחלקה מהווה חיבור בין 2 MazeNode ובכך מאפשרת את יצירת המבוך ביתר קלות.
* **MazeNode.cs** – מהווה עצם של חוליה במבוך. המחלקה כוללת בתוכה את המאפיינים של חוליה, כמו מי מוביל אליה, הריבוע המייצג אותה על המסך וגבולותיה. כמו כן, המחלקה כוללת פעולות האחראיות על יצירת החוליה על המסך.
* **OnlineOptionsWindow.xaml** – חלון המשמש חלון-מעבר לאחר שהמשתמש החליט לשחק נגד שחקן אחר. בו 2 אפשרויות – יצירת משחק, או הצטרפות למשחק.
* **OnlineOptionsWindow.xaml.cs** – הלוגיקה של חלון מרובה המשתתפים. יש בו בעיקר קריאה לפעולות והלוגיקה של הכפתורים עצמם.
* **Player.cs** – מהווה שחקן במשחק מרובה משתתפים. משומש בשביל ה-Host.
* **ScreenOrginizer.cs** – מנהל את כל לוגיקת יצירת המבוך והשמתו בחלון הראשי.
* **WindowInteraction.cs** – בו מוכלים כל החלונות הפעילים. המחלקה נועדה כדי שתהיה האפשרות ליצור חלונות חדשים ולסגור אותם, וכן לאפשר גישה קלה יותר לחלקים של כל חלון.

# תרשים זרימת המשחק

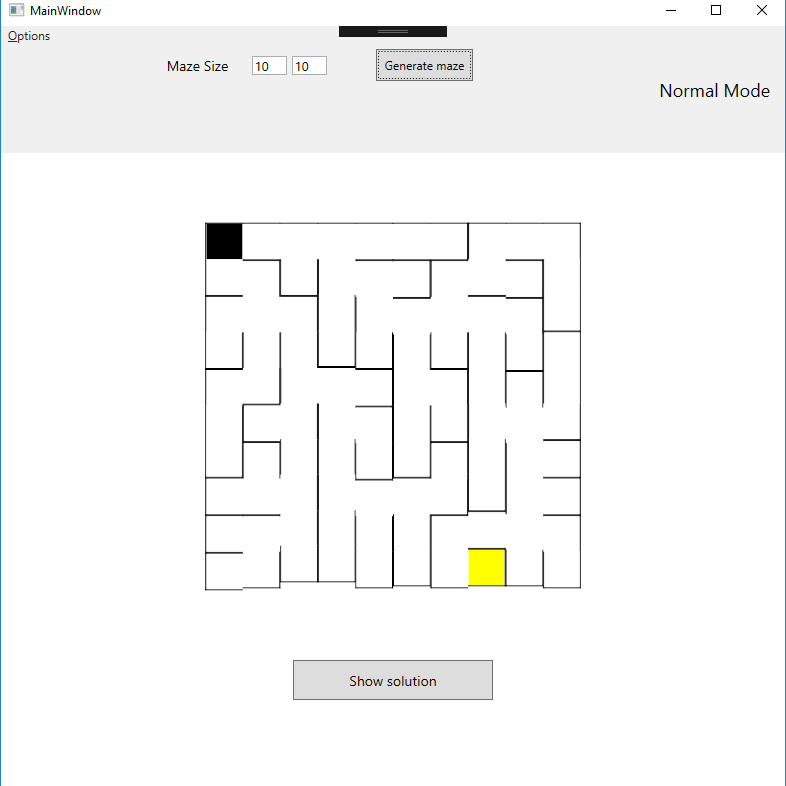
## מצב רגיל/מרובה משתתפים



## מצב משחק נגד השעון



# מבנה המסך



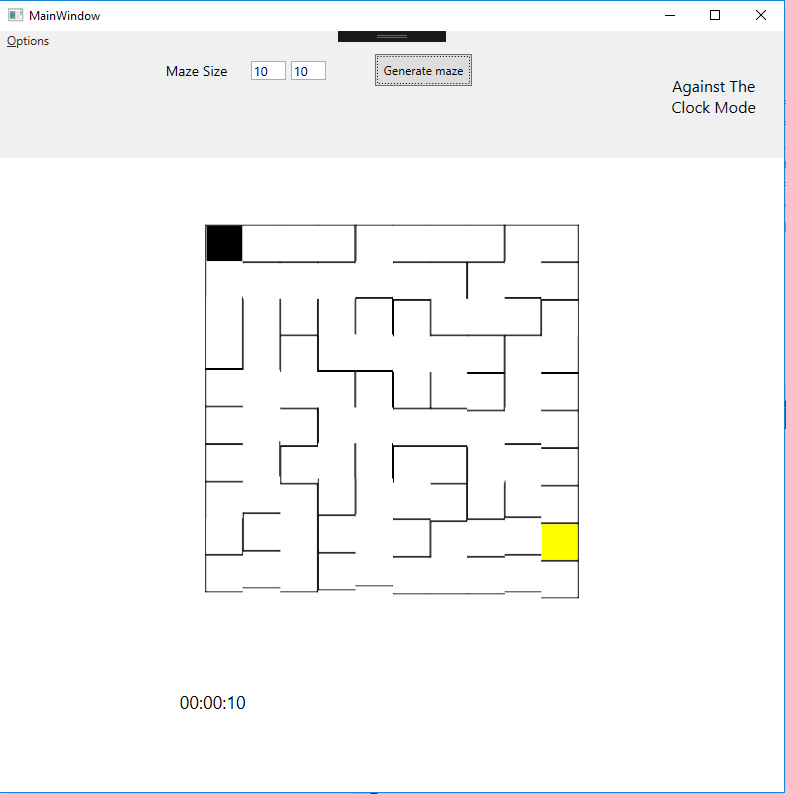
המסך כולל את תצוגת המבוך, נקודת ההתחלה תיצבע בשחור, נקודת הסוף בצבע צהוב

למעלה מופיעים תיבות הטקסט האחראיות לגודל המבוך, וכפתור ליצירת המבוך

בחלקו העליון של המסך מופיע תפריט המאפשר מעבר בין מצבים, ויציאה מהמשחק

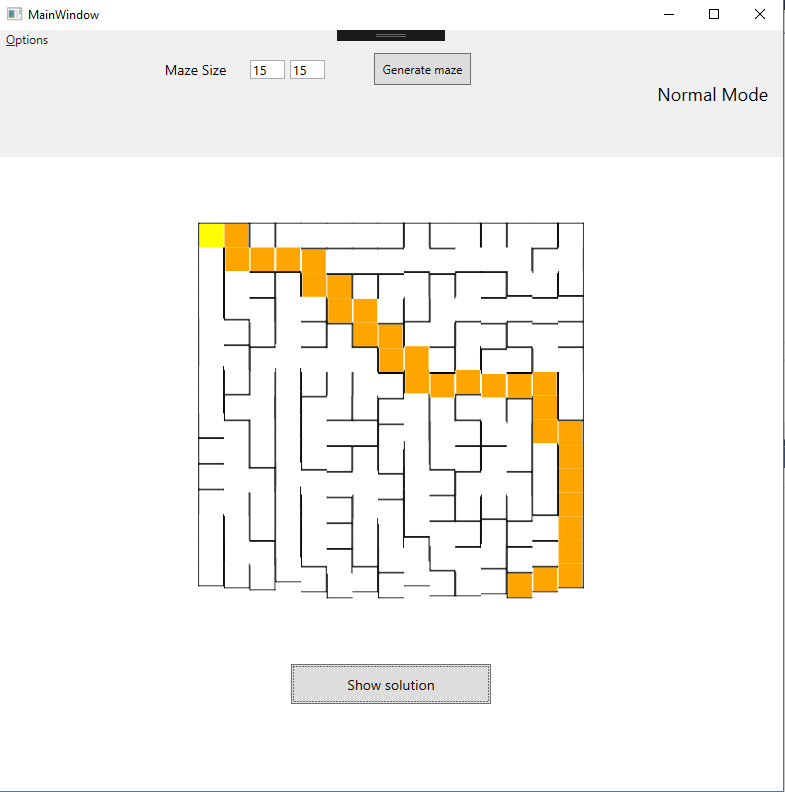
כמו כן, ישנו כפתור "מציאת הפתרון" שבעזרתו ניתן לראות את פתרון המבוך באופן מיידי

## מצב משחק נגד השעון



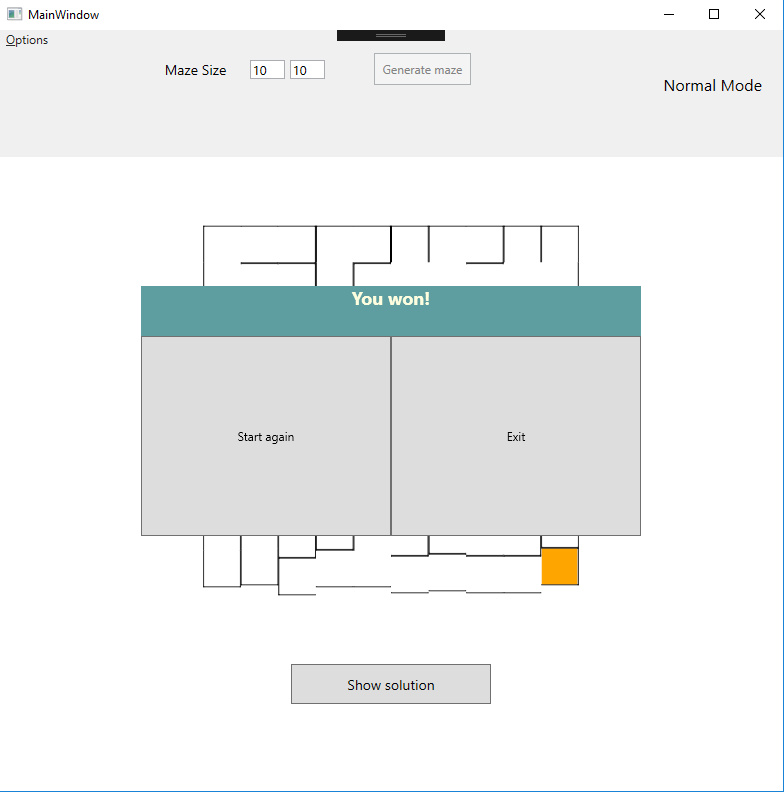
במצב זה הוסר כפתור מציאת הפתרון, ונוסף טיימר המחושב על פי גודל המבוך, הנועד לקבוע האם המשתמש הצליח לפתור את המבוך לפני הגעתו ל-0

## פתרון המבוך



לאחר שלחצנו על כפתור מציאת הפתרון, פתרון המבוך יוצג על ידי צביעת הריבועים הכוללים את הדרך אל קצה המבוך.

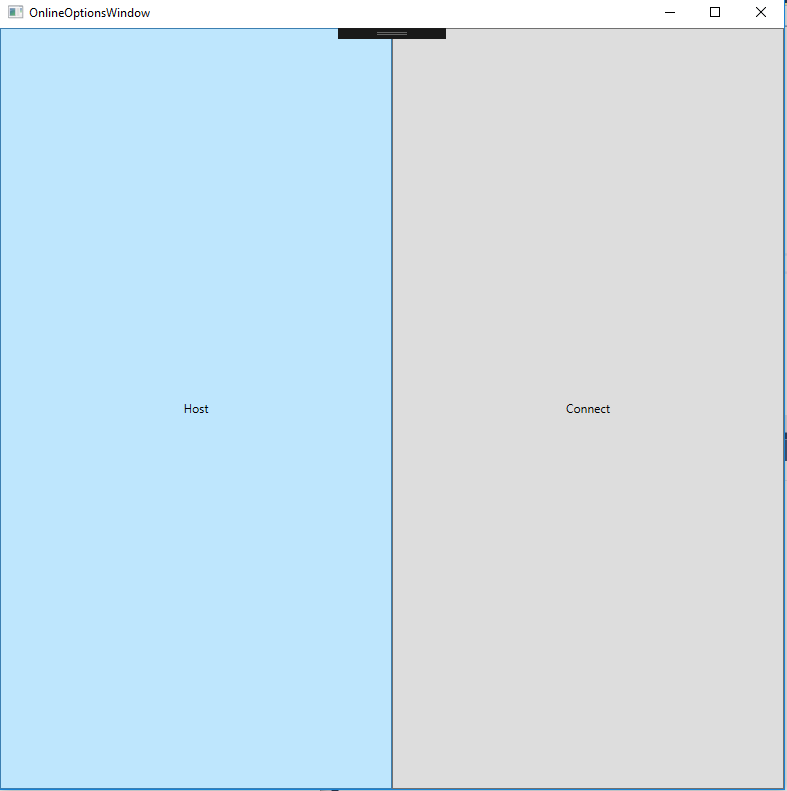
## מסך ניצחון/הפסד



כאשר המשתמש הגיע לקצה המבוך, יוצג בפניו חלון הניצחון, ובו אפשרות לשחק שוב, או לצאת מהמשחק.

במצב משחק נגד השעון, יוצג הכיתוב המעיד שהמשתמש הפסיד אם לא הצליח לסיים את המבוך בזמן שהוקצב עבורו.

## מסך אפשרויות מרובה-משתתפים



מסך מעבר המכיל 2 כפתורים, זאת כדי לקבוע האם המשתמש יוצר משחק/מצטרף למשחק קיים.

# תיאור המחלקות – פעולות מרכזיות

## אלגוריתם מרכזי



האלגוריתם המרכזי עובד בצורה הבאה:

הוא יוצר רשימה של חיבורים אפשריים לכל אחד מהשכנים של תא המקור, ולאחר מכן נכנס ללולאה שמתבצעת כל עוד ישנם חיבורים של תאים לטיפול. הלולאה מחברת באופן רנדומלי תאים חדשים לדרך הראשית של המבוך, ומסירה את כל החיבורים המובילים אל התאים שנוספו מהרשימה של החיבורים. לאחר מכן היא מוסיפה את כל החיבורים האפשריים לתאים חדשים לרשימת החיבורים, כאשר התאים שנוספים לא יכולים להיות חלק מהמבוך.

## תא במבוך



כל תא במבוך מורכב מכמה תכונות:

* השכנים שלו (במערך של 4 MazeNode)
* מי מוביל אליו
* האובייקט המוחשי שיופיע על המסך
* הגבולות שלו, שיקבעו על פי מי מוביל אליו וכן אם הוא נמצא באחד הקירות של המבוך

לאחר מכן, מוגדרות פעולות האחראיות ליצירת האובייקט שיופיע על המבוך, וגבולות התא

## יצירת התאים במבוך והגדרת השכנים שלהם



פעולה האחראית ליצירת מערך התאים במבוך, והגדרת השכנים של כל תא בהתאם למיקומו במערך

## טיפול בתזוזה





הפעולה אחראית על כל דימוי התזוזה במשחק. היא קולטת את לחיצות המקלדת של המשתמש על החצים, ולפיהם מדמה תזוזה על ידי צביעת המשבצות ובכך נראה כאילו המשתמש זז בתוך לוח המשחק.

## חיבור בין 2 חוליות של המבוך



מחלקה יחסית פשוטה אשר נועדה להקל על מציאת חיבורים בין 2 חוליות משחק.

משומשת בעיקר באלגוריתם המרכזי.

## השרת







השרת, שהוא בעצם ה-Host אחראי לשלוח את כל המידע של המבוך לשחקנים האחרים.

תחילה, הוא שולח את הגודל של המבוך, לאחר מכן את סדר החוליות, ולבסוף נותן את האישור להתחלת המשחק.

## הלקוח







הלקוח (Client) הוא משתמש אשר הצטרף למשחק קיים. קיימות בו פעולות לקבלת מידע מהשרת, כמו גודל המבוך, מבנה המבוך, וכן מה לעשות כאשר מתחיל המשחק.

# קשיים

במהלך שיעורי הסייבר היה לי את הזמן המתאים להשקיע בעבודה רצופה על הפרויקט, והצלחתי להגיע לרמה טובה ולהגשים את רוב המטרות. למרות זאת, במהלך העבודה נתקלתי בקשיים רבים, בעיקר מבחינת זמן:

* התקיימו בגרויות שונות שהקשו על הרצף התקין של פיתוח הפרויקט, ובמקרים רבים מצאתי את עצמי מנסה להיזכר במטרתי הנוכחית, וכן היה צורך בחידוד נושאים שונים שהספקתי לשכוח.
* נתקלתי בבעיות תכנותיות שונות שהיה צורך להשקיע עליהן זמן נוסף, ועל כן התכנון הכללי של הזמן שלי היה צריך להשתנות לטובת השקעת זמן נוסף עבור הפרויקט.
* חלק מהכלים בהם השתמשתי דרשו למידה רחבה, הבנתם וכן השימוש בהם – כמו Events, Threading ושימוש בצורות תקשורת.
* יצירת מצב מרובה משתתפים התברר כמסובך יותר משחשבתי, והייתי צריך ללמוד להשתמש ב-API שיצרה חברת תקשורת כדי לבצע את הפעולות כמו שצריך.

# מסקנות

גיליתי שכדאי לתכנן מבעוד מועד את מהלך העבודה, וכדאי לחשוב מראש כבר על בעיות שונות שיכולות לצוץ. תכנון מראש יאפשר יצירת לוח זמנים מתוכנן מראש, בו תהיה הקצאה לכמות הזמן שיש להשקיע על כל חלק, ויהיו פחות הפתעות

כמו כן, גיליתי שכדאי ללמוד היטב את הנושאים אותם אני הולך לממש בעבודה, ולא רק לרפרף עליהם ולנסות לממש אותם "על עיוור". ככה, יש הבנה עמוקה יותר בנושא, ובמקרה של טעויות, אפשר להבין בקלות איך להתגבר על הבעיה, ואף לתכנן מראש את דרך הפתרון לבעיה שתהיה ידועה מראש.

# ביבליוגרפיה

* **http://csharphelper.com/**

**באתר זה מצאתי דוגמה לכיצד אוכל לממש אפליקציה ב-WPF. בעזרת המבנה שהיה באתר הצלחתי לממש את המבוך בדרך שרציתי**

* **http://www.wpf-tutorial.com/**

**מאתר זה למדתי את מירב הכלים של WPF ובכך הצלחתי להשתמש בכלים הדרושים לכל דבר שרציתי לעשות בפרוייקט**

* **https://stackoverflow.com/**

**האתר המרכזי ממנו מצאתי תשובות לבעיות שונות שנתקלתי בהם במהלך הפיתוח**

* **http://www.networkcomms.net/**

**באתר זה מצאתי ספריה שאפשרה לי לבצע את התקשורת בצורה טובה ויעילה יותר, האתר כולל מדריכים שונים לשימוש בספריה, וכן API המכיל את כל הפעולות הדרושות**