**6 COLORS - README**

**תיאור:**

מטרת המשחק היא לצבור שטח גדול ככל האפשר.  
בתחילת המשחק, הריבועים שעל הלוח נצבעים באופן רנדומלי, ואז השחקנים מקבלים אוטומטית את הריבוע שבאחת הפינות: השחקן האנושי מתחיל בפינה השמאלית התחתונה (כלומר הוא שולט על הריבוע בפינה זו), והמחשב מתחיל מהפינה הימנית העליונה.  
בכל תור, השחקן בוחר צבע כלשהו הגובל בשטח שכבר שייך אליו, ובכך מגדיל את השטח שלו אל האריחים השכנים בעלי הצבע הזה. בכל שלב, מותר לבחור אחד מהצבעים הפנויים, כלומר כל הצבעים למעט שני הצבעים שנבחרו בשני הצעדים האחרונים על ידי השחקנים (וכרגע הם "תפוסים" על ידם). המנצח הוא השחקן הראשון ששולט על יותר מ־ 50% משטח הלוח.

**הסבר למשתמש:**

בתחילה מוצג המסך הראשי שכולל 4 אפשרויות:

1. **Single Player** – משחק עבור שחקן יחיד
2. **Create Game** – יצירת משחק עבור זוג משתתפים ברשת
3. **Join Game** – התחברות למשחק ברשת
4. **Exit** – כשמו.

כאשר בוחרים משחק עבור שחקן יחיד, ניתן לבחור 3 רמות:

1. **Easy –** משחק מול מחשב שפועל בשיטת "יותר מזל משכל" – המחשב מגריל תמיד צבע ומייחל לבחירה נכונה.
2. **Regular -** המחשב בוחר בצורה חמדנית. כלומר, הוא עובר על כל צורות "השפה" שלו, ובודק את צבע השכנים שלהם. הצבע שיזכה אותו במירב הנקודות - בו הוא יבחר.
3. **Super -** עובד גם בצורה חמדנית, אך עושה זאת בצורה 'עמוקה', ז"א בודק כמה באמת מתווסף לו בכל בחירה.

במשחק של שחקן יחיד, המשתמש תמיד מתחיל ראשון לשחק עם הצורה השמאלית התחתונה.

יצירת משחק תוביל למסך המתנה למחשב אחר ברשת שיתחבר אליו.  
במסך זה מופיעה כתובת ה-IP של המחשב. כאשר זוהה חיבור אז תוצג הודעה בהתאם, ואפשרות להתחיל את המשחק

התחברות למשחק קיים מובילה למסך החיבור למחשב שיצר את המשחק, יש להציג את כתובת ה-IP של המחשב השני, כפי שמופיעה במסך אצלו.

כאשר המחשבים התחברו למשחק, אז המחשב שיצר את המשחק הוא זה שתמיד יתחיל ראשון, והוא יתחיל עם הצורה השמאלית התחתונה.

מסך המשחק כולל 2 כפתורי תפריט - Restart ו-Exit, לוח המשחק, רשימה של 6 צבעים לבחירה ומידע על המשחק – אחוז שטח כיסוי לכל שחקן ושל מי התור כרגע.

בכל המסכים, מקש Esc חוזר למסך הקודם.

**מבנה פנימי של המשחק ברשת:**

התקשורת מתבצעת באמצעות ארכיטקטורת שרת-לקוח.לשם כך, השתמשנו בספריית Network של SFML.  
כמו כן, המשחק ברשת כולל 2 תהליכונים: אחד עבור הרצת המשחק והשני עבור ניהול התקשורת. המימוש נעשה ע"י std::thread  
התהליכונים מתקשרים בניהם באמצעות תורי בקשות מסונכרנים (תור שליחה ותור קבלה) שמוגנים ע"י הסמפור קריאה/כתיבה std::shared\_mutex.מיד לאחר ההתחברות נשלח מספר טבעי מהשרת ללקוח שמטרתו לסנכרן את הגרעין של מחולל המספרים האקראיים של הספרייה לאותו דבר, ובכך יוצא שהלוחות מוגרלים בצורה זהה, כי סדרת המשחק המוגרלים תהיה זהה.  
בכל תור שהתבצע, פעם מהשרת ופעם מהלקוח, נשלח מספר טבעי שמהווה את הצבע RGB שנבחר, וככה המשחק מתנהל עד לסיום.  
אם שחקן מתנתק באמצע המשחק או שאירע כשל בשליחת המידע – תוצג הודעת שגיאה בהתאם.

**רשימת קבצים:**

נחלק בין קבצי המשחק לבין קבצי ה-GUI:

**קבצי המשחק:**

|  |  |
| --- | --- |
| שם הקובץ | הסבר קצר |
| PlayerBase | מחלקת בסיס אבסטרקטית המשמשת לייצוג שחקן במשחק. |
| UserPlayer | יורשת מ- .PlayerBase המחלקה בה משתמש שחקן אנושי. |
| PlayerAI | מחלקת בסיס אבסטרקטית המשמשת לייצוג שחקן מחשב. |
| PlayerAIStupid,  PlayerAIRegular, PlayerAISuper | יורשים מ- PlayerAI כל אחת מייצגת רמת קושי אחרת של מחשב. |
| MainScreen | המסך הראשי. יורש BaseScreen<RelativeLayout>.  כולל 4 כפתורי תפריט. |
|  |  |
| ChooseAIModeScreen | מסך בחירת רמת קושי. 3 רמות. יורש BaseScreen<RelativeLayout> |
| GameScreen | יורש BaseScreen<RelativeLayout>  מסך המשחק. מכיל פאנל עליון של כפתורי יציאה\RESTART. מכיל את לוח המשחק. וכן פאנל תחתון של בחירת הצבעים |
| WaitingMultiplayerScreen | יורש BaseScreen<RelativeLayout>  מסך המתנה להתחברות של מולטיפלייר |
| JoinGameScreen | יורש BaseScreen<RelativeLayout>  מסך הצטרפות למשחק ברשת. |
| WinScreen,  LoseScreen | יורש BaseScreen<RelativeLayout>  מסכי ניצחון/הפסד |
| ColoringAlgorithem | אלגוריתם צביעה של הלוח |
| Board | לוח המשחק. יורש RelativeLayout<PolygonView> |
| LevelDifficultyButton | כפתור רמת קושי. יורש ImageButton |
| AreaButton | יורש .TextView כפתור המציג אחוזי שטח. (יש לשחקן ויש ליריב) |
| ColorButton | יורש Button. כפתור צבע |
| ColorPanel | יורש HorizontalLayout<ColorButton>  מכיל מערך של הצבעים ואת כפתורי הצבעים |
| BottomPanel | יורש HorizontalLayout<View> מכיל את פאנל הצבעים וכפתורי השטח |
| GameController | מנהל את מהלך המשחק. מריץ את מסכי המשחק. |
| Shape | מחלקה אבסטרקטית של צורה. מכיל גובה, רוחב ומס צלעות. |
| Square | יורש shape. ריבוע |
| Triangle | יורש Shape. משולש. מכיל enum של ארבעת הכיוונים של המשולשים. |
| PolygonShape | מצולע. יורש Shape |
| PolygonView | מחלקה לתצוגת מצולע. יורשת View. המצולע חייב להיות קמור ושסדר הוספת הנקודות שלו תהיה כולם בכיוון השעון או נגד שעון כיוון השעון. |
| Utilities | מרחב שם עם פונקציות עזר כגון בחירת צבע רנדומלי |
| GameMenu | תפריט עליון של מסך המשחק. יורש HorizontalLayout<View> |
| Graph<T> | גרף לא מכוון. מכיל רשימת קדקודים שקדקוד היא מחלקה מקוננת בתוך הגרף שמחזיקה אובייקט T. כל קדקוד מכיל רשימת שכנים. |
| INetworkThread | ממשק תקשורת כללי. מכיל תהליכון של התקשורת. |
| Matrix<T> | מטריצה. כוללת אפשרות לאיטרטור. |
| NetworkException | חריגת תקשורת. יורש std::exception |
| NetworkPlayer | שחקן ברשת. יורש PlayerBase |
| RequestsClientThread | מחלקת ניהול תקשורת של הלקוח ע"י תורי בקשות |
| RequestQueue | מעטפת של תור עם סמפור סנכרון גישה. |
| RequestsServerThread | מחלקת ניהול תקשורת של השרת ע"י תורי בקשות |
| ServerPlayer | שחקן השרת. נמצא אצל הלקוח. כלומר, הלקוח משחק מולו. יורש NetworkPlayer |
| ClientPlayer | שחקן הלקוח. נמצא אצל השרת. יורש NetworkPlayer |
|  |  |
|  |  |

**קבצי ה-:GUI**

|  |  |
| --- | --- |
| שם הקובץ | הסבר |
| View | מחלקה מופשטת המייצגת אובייקט שמוצג על המסך. כוללת מיקום, רוחב, גובה, רקע (תמונה או צבע), גבול (צבע וגודל) כוללת אירועים: לחיצה, כניסה ויציאה עם העכבר. לחיצה ועזיבת מקש במקלדת. פונקציה לציור, קבלת מלבן חוסם, מצב תצוגה/הסתרה, מצב אפשור/ביטול אירועים. אפשרות למצלמה.  לא ניתן ליצור מופע של המחלקה באופן ישיר אלא ע"י ירושה בלבד. |
| TextView | מחלקה המייצגת טקסט. ניתן לשנות את הפונט, הטקסט וצבעו, ואת כיוון היישור שלו (ימין, מרכז או שמאל) |
| Button | כפתור. יורש TextView |
| AnimationView | אנימציה. יורש View. מכיל TextureInfo מקושר וטיימר, ובכך מצייר אנימציה. כולל אופציה לשינוי תדירות עדכון התמונות. |
| ViewGroup<ViewType> | מחלקה מופשטת תבניתית המכילה וקטור של אובייקטים ViewType. ViewType חייב להיות צאצא של View  מטרת המחלקה לגרום לכך שכל אירוע שמתקבל יגיע לאובייקטים המוכלים ב- ViewGroup  כמו כן, גם מצב התצוגה והאפשור מועבר לילדים. כלומר, לאובייקטים המוכלים בו.  לא ניתן ליצור מופע של המחלקה באופן ישיר אלא ע"י ירושה בלבד. כולל הנחה שאין אובייקטים מצטלבים (הסיבה היא בגלל חישוב יעיל של האירועים) |
| VerticalLayout<ViewType> | שכלול של ViewGroup שמסדר את האובייקטים בצורה אנכית אחד אחרי השני. גודל האובייקטים מחושבים באופן יחסי בהתאם לממדי ה- VerticalLayout |
| HorizonalLayout<ViewType> | שכלול של ViewGroup שמסדר את האובייקטים בצורה אופקית אחד אחרי השני. גודל האובייקטים מחושבים באופן יחסי בהתאם לממדי ה- HorizonalLayout |
| RelativeLayout<ViewType> | שכלול של ViewGroup שמסדר את האובייקטים באופן מותאם אישית. גודל האובייקטים מחושבים באופן יחסי בהתאם לממדי ה- RelativeLayout |
| ImageView | אובייקט המייצג תמונה. יורש View |
| ImageButton | שכלול של VerticalLayout המכיל ImageView ו-TextView |
| ImageTexture | מכיל מידע על תמונה או צבע ומצייר אותה בהתאם למלבן חוסם שניתן לו |
| Border | גבול – מכיל צבע ועובי הגבול ומצייר אותו בהתאם למלבן חוסם שניתן לו. |
| EditText | עורך טקסט. יורש TextView. |
| RectUtils | פונקציות עזר עבור מלבנים |
| IHandler<T> | מנגנון אירועים גנרי תבניתי. מכיל וקטור של פונקציות שמאזינות לאירוע מסוים. כולל פונקציות הוספת האזנה וקריאה לאירוע. |
| Cell | תא – מכיל מספר שורה ומספר עמודה |
| Matrix<T> | מטריצה תבניתית. כוללת תמיכה ב-iterator |
| TextureInfo | תמונה/אנימציה. מכיל Texture שמייצג תמונה או סדרת תמונות ומטאדטה נלווה – שם, מספר שורות, עמודות ומספר תמונות. |
| ResourceLoader | מחלקה מופשטת סינגלטונית לטעינת משאבים |
| TextureManager | טוען תמונות/אנימציות מקובץ JSON. יורש ResourceLoader |
| SoundManager | טוען צלילים של המשחק ומוסיקת רקע מקובצי JSON. יורש ResourceLoader |
| FontManager | טוען גופנים מקובץ JSON. יורש ResourceLoader |
| LoadResourceException | חריגה בעת טעינת משאב מסוים. יורש std::exception |
| Timer | טיימר. מחזיק שעון עם פונקציה שמופעלת כל מרווח זמן.  פונקציות התחלת שעון וסיום שעון. ובדיקה של השעון. משתנה בוליאני האם השעון פעיל. |
| StopWatch | שעון עצר. מחזיק שעון עם פונקציה שמופעלת כאשר הטיימר יסיים. כולל אירוע בדיקה כל מרווח זמן. |
| BaseScreen<ViewType> | מחלקה אבסטרקטית של מסך. יורש ViewType. |

**מבני נתונים עיקריים:**

השתמשנו במטריצה כדי לצייר את הצורות ולקבוע את השכנים.

בשאר המשחק, הלוח מיוצג על ידי גרף המורכב מוקטור שמחזיק את רשימת הצורות וכל צורה יש רשימת שכנים שנמצאת ב unordered\_set.

**אלגוריתמים הראויים לציון:**

* AI Super - משתמש ברקורסיה ע"מ למצוא את הרווח האמתי שלש כל בחירת צבע.
* AI Regular - עושה בחירה חכמה, אך בסיסית יותר.
* ColoringAlgorithem- מקבל סט של קודקודי משתמש וקודקודי "גבול" (= קודקודי משתמש שנוגעים בקדקודים שאינם של המשתמש),עובר על רשימת הגבול ,מצרף בעזרת רקורסיה את כל מי שאינו שייך למשתמש אך באותו צבע של המשתמש ומעדכן את רשימת הגבול.

**ספריות צד ג':**

* [**nlohmann - json**](https://github.com/nlohmann/json)

**הערות:**

* במשחק עם זוג משתתפים אין כפתור restart. במידה ורוצים משחק חדש יש להתחבר מחדש ע"י Create Game ו-Join Game.
* הערות על בעיות שעלולות לקרות בהתחברות למשחק ברשת (מניסיון כואב ומלמד..):  
  1. ה-WIFI של המכללה (ולא קו ה-Ethernet) חוסם חיבורים ברשת המקומית, אז אין למה לבדוק דרכו...  
  2. יתכן שהפורט באחד המחשבים כבר תפוס וזה גורם לכשל ברשת. הפורט במשחק מוגדר כקבוע בממשק התקשורת INetworkThread, (לשים לב שב-2 המחשבים הפורט חייב להיות אותו הדבר)  
  3. לשים לב שמדובר על רשת מקומית בלבד ולא על רשת האינטרנט.

1. בפעם הראשונה שיוצרים משחק, יפתח חלון של חומת אש של וינדוס שיבקש הרשאת גישה למשחק להאזנה לפורט ברשת. יש לאשר כדי שהחיבור יצליח.

* במחלקות השונות יש פונקציה toString() שמיועדת לצורך דיבאגינג בלבד.
* בשביל הנוחות, ניתן לשנות את גודל המסך. מומלץ להגדיל אותו למקסימום, כי הלוח המוגרל כולל הרבה צורות (50\*50)

**מגישים:**  
מתנאל אביוף  
נחום נגה  
עקיבא גולדשטיין