תכנות מתקדם 2 – ממשק משתמש (GUI)

תאריך הגשה: 13.04.2020 – נא להגיש במערכת submit, התרגיל ניתן להגשה בזוג.

רקע

אנחנו רוצים לכתוב ממשק משתמש נוח לתפעול של מטוס קטן. המטוס שלנו טס בתוך סימולטור טיסה (Flight Gear), ומגיב להוראות הנוגעות להגאים ולמשטחי השליטה השונים: Rudder – הגה כיוון, Elevator – הגה גבה, Aileron – מאזנות, Throttle – מצערת.

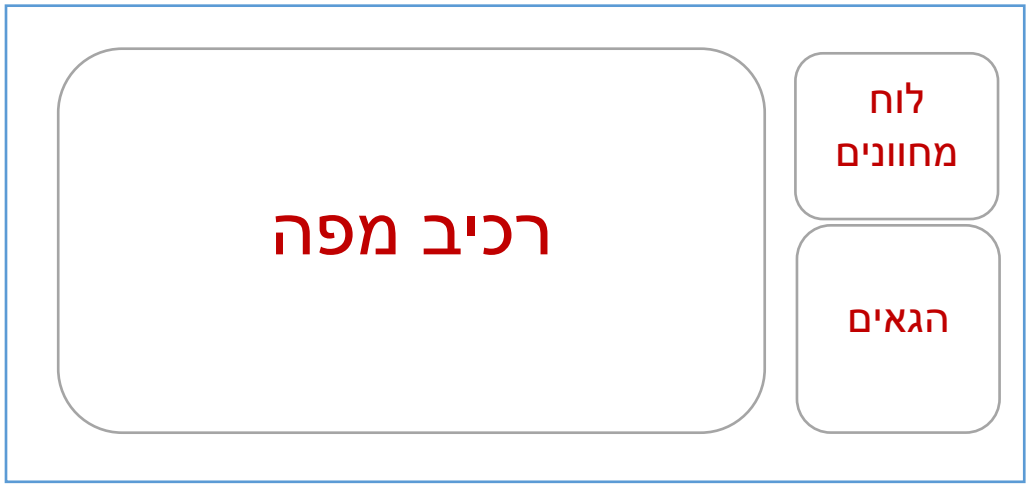
סימולטור הטיסה המקורי עובד עם ממשק גרפי תואם לתא הטייס ונועד לאימון טייסים, אנחנו נבנה ממשק פשוט יותר לשליטה על הסימולטור. האפליקציה שלנו תפתח ערוץ תקשורת ייעודי מול הסימולטור בו נעביר פקודות לשינוי מצב ההגאים או לקבלת מידע מן החיישנים המותקנים במטוס.

למידה עצמית

התרגיל הזה, כמו יתר התרגילים בקורס מתבסס ברובו על למידה עצמית של סביבות עבודה וטכנולוגיות לא מוכרות, ומיועד להקנות לכם דרכי התמודדות עצמאית עם בעיות בתהליך הפיתוח. התרגיל כתוב כמסמך דרישות לא מאוד טכני, אלא מנקודת המבט של לקוח כלשהו. חלק מן הדרישות שתקבלו אינן מפורטות במכוון, ומצופה מכם למצוא את הפתרון שעונה נכון ובאופן סביר על צרכי הלקוח. אין שום מניעה להתחיל לעבוד על התרגיל גם אם לא למדנו עקרון מסוים או טכנולוגיה מסוימת.

טיפ – תעברו קודם על ה- TUTORIAL הבא: walkthrough-my-first-wpf-desktop-application

דרישות פונקציונליות



1. האפליקציה יודעת לדגום את הסימולטור בצורה תדירה על מנת להציג על גבי מפה את מיקום המטוס בעולם, וכן לעדכן ביעילות את מיקום המטוס עם כל שינוי בהגאים.
2. בנוסף, האפליקציה מציגה למשתמש בחלון ייעודי נתונים שונים הנדגמים מחיישני המטוס.
3. רכיב ההגאים כולל בתוכו ג'ויסטיק ואלמנטים נוספים לשליטה על ההגאים הראשיים.
4. **ה- UI לעולם לא נתקע**, גם אם לוקח לסימולטור הרבה זמן להגיב. (ניתן לפתוח Thread או Task שמתשאל את הסימולטור ומעדכן א-סינכרונית את ה- UI).
5. **האפליקציה לעולם לא מתרסקת** כאשר נתקלת בבעיה, אלא מציגה חיווי שאינו מפריע לעבודה השוטפת, על סוג או מהות הבעיה למשתמש. יש לזהות ולדווח על מקרי קצה מוכרים בחיווי זה.

ממשק מול הסימולטור

הסימולטור עובד עם פקודות טקסטואלית מהצורה: [get/set] <variable-name> ומחזיר ערך טקסט

set /controls/engines/current-engine/throttle 0.5 ➔ 0.5
set /controls/flight/rudder -20 ➔ -1
get /position/latitude-deg ➔ 32.1093
get /instrumentation/gps/indicated-ground-NOT-VARIABLE-NAME ➔ ERR

הפקודות נשלחות באמצעות ערוץ תקשורת מעל גבי פרוטוקול TCP/IP. כתובת ה- ip ומספר ה- port יהיו מסופקים בברירת מחדל על ידי מנגנון הגדרות App.config של NET. וניתנים לשינוי באמצעות פקד ייעודי.

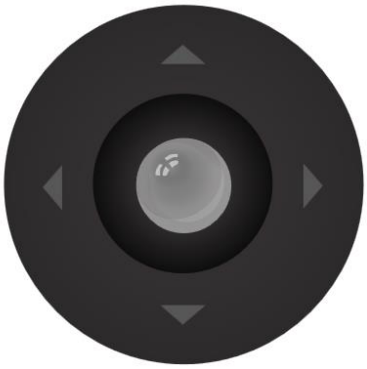
- בערוץ זה הפקודות יישלחו בצורה טקסטואלית ויוחזר קלט בצורה טקסטואלית.
  למשל אם למשתנה מסוים יש ערך 0.95 אז פקודת get תחזיר את המחרוזת: 0.95
- לאחר פקודת set יוחזר ערך המשתנה החדש כטקסט.
- אם הערך המספרי בפקודת set חורג מטווח הערכים המותר של המשתנה, תתפרש הפקודה כהשמה של הערך הקרוב ביותר לטווח הערכים – כלומר אחד מקצוות הטווח. (בדקו את docs של flight-gear)
  למשל בפקודה set X 10000 כאשר הטווח המותר של X הוא [1 -1] המשתנה X יקבל את הערך 1.
- בכל בעיה או שגיאה בפקודה תוחזר מחרוזת (string) עם הערך: ERR.
- למען מהירות תהליך הפיתוח נעבוד עם מדמה לסימולטור אשר יממש את הממשק המצוין לעיל, ולא נעבוד ישירות עם Flight Gear. קוד לדוגמא של המדמה יפורסם בהמשך ונועד לבדוק את תקינות הערוץ.

רכיב מפה

- הסימולטור מחזיר את מיקום המטוס כקווי אורך וגבה. קריאת רקע:
  https://en.wikipedia.org/wiki/Geographic_coordinate_system
  רכיב המפה יציג את מיקום המטוס על גבי מפה דו-מימדית. אם המטוס מעל נקודת ציון מסוימת (קו אורך, קו גבה) יש להציג סימן גרפי על גבי המפה לציון המקום.
- מיקום המטוס יידגם בתדירות וביעילות מהסימולטור באמצעות הממשק שתואר לעיל. על רכיב המפה להגיב במהירות לשינויים בהגאים ולהציג למשתמש את המיקום החדש בצורה מהירה.
- קו האורך: /position/longitude-deg קו הרוחב: /position/latitude-deg
- ניתן (ואף רצוי) להשתמש ברכיב מפה מוכן למשל: Bing Maps WPF Control ניתן לבחור ברכיבים מוכנים אחרים שיש לשלב בתוך האפליקציה שלכם.
- ניתן לממש את רכיב המפה גם על ידי מפה כתמונת רקע סטטית. עליכם להמיר את הקואורדינטות המוחזרות על ידי הסימולטור:Latitude and Longitude למיקום (פיקסל) על גבי המפה. ניתן להיעזר בספריות שונות, למשל:https://archive.codeplex.com/?p=proj4net
  אופציה נוספת היא ללמוד כיצד ממירים בעצמכם באמצעות הנוסחאות הללו:
  https://en.wikipedia.org/wiki/Mercator_projection#Derivation_of_the_Mercator_projection
- כל האופציות למימוש רכיב מפה הן מאתגרות ומצריכות להתמודד עם רכיבים וספריות חיצוניים או לרכוש ידע על נושא שאינו נלמד בקורס, אך קשור לעולם התוכן של האפליקציה שאנחנו בונים.
- ה – dll החיצוני (שלא יצרתם) **היחיד** שמותר לכם לשלב באפליקציה, מלבד תשתית Net., יהיה שייך לרכיב המפה.

רכיב ההגאים

קיבלתם ממעצב גרפי עיצוב של רכיב שליטה במטוס בתצורה של ג'ויסטיק יחד עם קוד מינימלי שאחראי בעיקר לאנימציה המחזירה את הידית למרכז. יש להשתמש בפקד זה ולהוסיף לו בקוד את הפונקציונליות הבאה:



- המשתמש יזיז את הידית על ידי לחיצה וגרירה של הידית באמצעות העכבר.
- כשהמשתמש משחרר את לחיצת העכבר הידית חוזרת למרכז (מצב מנוחה) על ידי הפעלת האנימציה המצורפת.
- כאשר הידית בתנועה ההיסט בפיקסלים של מיקום הידית (ממרכז הידית במצב מנוחה) מתורגם לקואורדינטות יחסיות X,Y בטווח של [1 1-] כאשר המיקום 0,0 מציין את מרכז הג'ויסטיק, והמיקום [X=1, Y=0] מציין שהידית בקצה האופקי הימני ביותר (אך אנכית היא אפס). בדומה המיקום [X=0, Y=1] מציין את הקצה העליון.
- כל שינוי (תנועה) במצב הידית (X, Y משתנים) מתורגם מיידית לפקודה להגאים הרלוונטיים: הגה כיוון (rudder) עבור X, והגה גבה (elevator) עבור Y. הבהרה: כל עוד הידית נלחצת ונגררת, מצב ההגאים שונה מאפס-אפס.
- כאשר הלחיצה משוחררת, והידית חוזרת למרכז (למצב X=0, Y=0), יש לדאוג לאפס את ההגאים.
- מרכז הידית אינו יכול לצאת מגבולות הג'ויסטיק (העיגול החיצוני השחור-אפור).
- יש ליצור פקד חדש המתבסס על העיצוב הקיים של הג'ויסטיק אשר מאפשר שליטה על כל ארבעת ההגאים. הגאי גבה וכיוון באמצעות הג'ויסטיק, מצערת ומאזנות באמצעות רכיבים אחרים, למשל: סליידרים. הפקד החדש מציג את ערכי כל ארבעת ההגאים ישירות, ללא תלות בערך בסימולטור, וכן שינוי בפרד של מצערת או במאזנת יעדכן את הסימולטור בהתאם.

רכיב לוח מחוונים

לוח המחוונים צריך לקבל באופן שוטף מידע מסימולטור הטיסה ולהציגו למשתמש. להלן רשימת התכונות הנחוצות:

1. indicated-heading-deg
2. gps_indicated-vertical-speed
3. gps_indicated-ground-speed-kt
4. airspeed-indicator_indicated-speed-kt
5. gps_indicated-altitude-ft
6. attitude-indicator_internal-roll-deg
7. attitude-indicator_internal-pitch-deg
8. altimeter_indicated-altitude-ft

מקרי קצה

- האפליקציה צריכה להגיב למצבי קצה כמו הסימולטור לא פועל ברקע או לא זמין משלל סיבות, או מחזיר קלט או פלט שאינם לפי הפרוטוקול המוגדר. **האפליקציה בשום פנים ואופן לא צריכה להתרסק**, אלא להתעלם מהקלט הנ"ל. צריך להוסיף חיווי (לבחירתכם) שאינו מפריע למשתמש על סוג הבעיה.
- לתמוך מצב קיצון אפשרי הוא שהמטוס טס מעל הקוטב הצפוני והתצוגה צריכה להתעדכן בהתאם.
- נקודות ציון מחוץ לגבולות כדור הארץ הם שגיאה של הסימולטור, וצריך להתמודד בהתאם.
- לסימולטור לוקח יותר מ-10 שניות להחזיר תשובה לפקודת get או set.

Deployment

- האפליקציה צריכה לרוץ בסביבה בה מותקן windows 10 עם גרסה עדכנית של .NET Framework 4 (גרסא 4.6 ומעלה). הקומפילציה צריכה לעבוד על visual studio 2019.
- כל המשאבים (config, dll) של האפליקציה צריכים להיות self-contained. אם אתם משתמשים בחבילות נוספות יש לדאוג שהם יצורפו לאפליקציה בתיקיית ההרצה (סעיף הבא).
- הקוד שלכם צריך להתקמפל ללא **שגיאות וללא אזהרות** לתוך תיקייה שמכילה את כל הקבצים הנחוצים לריצה, כולל dll חיצוניים או כל משאב חיצוני (תמונה) שהשתמשתם בו.
- שם ה- Solution: FlightSimulator שם ה- Project: FlightSimulatorApp
- התיקייה הנ"ל תהיה ברמת ה- solution ותיקרא out. קובץ ההרצה יושב בנתיב הבא: [...]\FlightSimulator\out\FlightSimulatorApp.exe

בדיקה וציון

- ככלל, זכרו: בודק התרגילים הוא הלקוח שלכם ואין לו סבלנות למוצר שלא עובד בלחיצה אחת.
- אם האפליקציה לא מתקמפלת ב- Visual Studio 2019 היא תוחזר לסטודנט להגשה מחודשת עם הורדה בציון של 10 נקודות. ניתן להגיש מחדש עד 24 שעות ממועד ההחזרה.
- בהמשך יסופק סקריפט אשר מקמפל אוטומטית ומריץ את האפליקציה לנוחיותכם.
- הציון מורכב מ-3 חלקים: פונקציונליות בסיסית (60%) עמידה במקרי קצה (10%) ושימוש ב- design pattern ידועים וסטנדרטים שלמדנו (30%)
- כל הדרישות הפונקציונליות צריכות להיות ממומשות ללא באגים הנראים לעין בשימוש ראשון (ללא בדיקת מקרי קצה וחריגים) התרחיש הסביר בו מטוס טס מעל נתב"ג ומסלול המטוס מתעדכן עם שינוי בהגאים **צריך להיות ממומש ללא רבב**. במידה ותרחיש הבסיס לא עובד יורדו עד 60 נקודות מהציון (לשיקול הבודק בהתאם לרמת הכשל).
- אי עמידה במקרי הקצה תוביל לירידה של עד 10 נקודות מהציון.
- איכות קוד: אתם נדרשים לכתוב קוד על ידי שימוש בתבנית MVVM שלמדנו. עליכם לממש 3 פקדים חדשים ולהוסיף את הפונקציונליות הנדרשת לפקד הג'ויסטיק. הפקדים צריכים לכלול רכיב ViewModel אשר אינו מודע לקיומו של ה- View. עליכם לשאוף שניתן יהיה להריץ את האפליקציה שלכם על ידי סימולציה של ה- View (מחלקה אחרת ללא ייצוג גרפי שמקפיצה אירועים ל-VM על כפתורים שנלחצו).
- בנוסף, יש לכתוב לפי ה – Code Conventions המומלצים של #C. להלן כללים בסיסיים: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions
- אי-עמידה בדרישות איכות הקוד תגרור הורדה של עד 30 נקודות מהציון הסופי לשיקול הבודק.

הוראות הגשה

- יש להגיש במערכת submit עד 23:59 ביום ההגשה.
- הוסיפו קובץ etc.txt ברמת ה- solution בפורמט id, full name למשל:

```
123456789,Paul R
987654321,Daniela S
```

- יש להגיש את כל קבצי הקוד xaml. cs. וכן את קבצי הפרויקט csproj. sln. אם השתמשתם ב- dll חיצוניים עבור רכיב המפה, או בקבצים אחרים יש לצרף אותם גם.
- בצעו clean solution וודאו שכל הקבצים הנדרשים לקומפילציה והרצה קיימים בתיקיית ה- solution.
- בצעו zip לתיקיית ה-solution. העתיקו את ה- zip לתיקייה אחרת ופתחו, וודאו שמתקמפל ורץ
- יש להגיש את FlightSimulator.zip תיקיית ה- solution ב- zip.
- ה – dll החיצוני (שלא יצרתם) **היחיד** שמותר לכם לשלב באפליקציה, מלבד תשתית .Net, יהיה שייך לרכיב המפה. **בהצלחה**