מטלה – דגמים והתנגשויות

א. שינוי ושיפור המשחק מהשיעור

הורידו מגיטהאב את הקוד של משחק-החלליות שבנינו בהרצאה (זה כאן https://github.com/erelsgl-at-ariel/gamedev-5780-code בתיקיה 2, הסצינה עם המספר הכי גבוה – 3d-shield).

שימו לב: אין להוריד בעזרת zip אלא בעזרת git clone. כמו כן כדי לחסוך זמן מומלץ להגביל את העומק:

```
git clone --depth=1 https://github.com/...git
```

ודאו שאתם מבינים את הקוד, ובצעו לפחות שינוי אחד מתוך הרשימה הבאה:

1. המגן לא נמצא על המסך בהתחלה, אלא מופיע מדי-פעם בנקודה אקראית. כשהשחקן מתנגש במגן, נוסף עיגול מסביב לחללית של השחקן. צבע העיגול נחלש משניה לשניה עד שהוא נעלם אחרי 5 שניות.
2. מדי-פעם מופיע על המסך תותח, בנקודה אקראית. כשהשחקן אוסף את התותח, הוא יכול לירות לייזר גדול וחזק יותר, למשך מספר שנית. התותח הוא חד-פעמי כמו המגן – נעלם אחרי שהשחקן אוסף אותו [אם אתם מסתבכים, תוכלו למצוא הסבר בגיטהאב של הקורס בתיקיה 6. אבל תנסו קודם לבד].
3. החללית של השחקן לא נהרסת מייד כשהוא מתנגש באויב, אלא יש לו בתחילת המשחק 3 "נקודות פגיעה" (hit points), כל פגיעה באויב מורידה לו נקודה אחת, ורק כשהוא מגיע לאפס הוא נהרס.
4. השחקן לא יכול לירות לייזרים בלי הפסקה, אלא חייב לחכות זמן מסויים (נניח חצי שניה) בין יריה ליריה הבאה.
5. יש שני סוגי אויבים: אחד איטי וכשפוגעים בו מקבלים נקודה אחת, והשני מהיר וכשפוגעים בו מקבלים שתי נקודות (כיתבו קוד כללי שיהיה קל להרחבה לשלושה סוגי אויבים או יותר).

בנוסף לשינוי מהרשימה, הוסיפו לפחות שינוי אחד מקורי.

הגשה:

- העלו את המשחק עם השינויים שהוספתם ל itch.io. **ודאו שהמשחקים במצב public**.
- העלו את הקוד לגיטהאב והוסיפו קובץ README.md המסביר מה בדיוק עשיתם, כולל הפניות לקוד. **ודאו שהתיקיה שאתם מעלים כוללת את הקבצים gitignore, gitattributes כמו בגיטהאב של הקורס, ושאתם לא מעלים את הקבצים הזמניים של יוניטי.**

ב. גבולות

כזכור, אחד הרכיבים הרשמיים בכל משחק הוא **גבולות**. כברירת-מחדל, במשחקים של יוניטי העולם הוא אינסופי ושטוח. הראו איך להשתמש בקוליידרים כדי ליצור גבולות במשחק החלליות, כך שהעולם יהיה:

1. עולם שטוח עם גבולות גלויים, כגון קירות שאי-אפשר לעבור;
2. עולם שטוח עם גבולות בלתי-נראים; לדוגמה, כשהאויבים עוברים את החלק התחתון של המסך, הם נהרסים; כשהלייזר עובר את החלק העליון של המסך – הוא נהרס.
3. עולם עגול – כשהשחקן מגיע לצד אחד של העולם, הוא מופיע בצד השני.

הגשה:

- העלו את אחד המשחקונים שיצרתם ל itch.io.
- העלו את הקוד לגיטהאב והוסיפו קובץ README.md המסביר מה בדיוק עשיתם, כולל הפניות לקוד.

ג. מימוש תהליכי-ליבה קלאסיים

כזכור, **תהליך-הליבה** הוא אוסף הפעולות המתבצעות שוב ושוב במהלך המשחק. הנושאים שלמדנו בשיעור הזה מאפשרים לכם כבר לממש תהליכי-ליבה של הרבה משחקים קלאסיים. ממשו את תהליך-הליבה באחד מהמשחקים הבאים לפי בחירתכם:

1. Jumper frog – צפרדע צריך לעבור את הכביש בלי להיפגע ממכוניות. *הרעיון*: צרו שלושה מסלולים שידמו את הכביש, וצרו spawner לכל נתיב שמתזמן את האובייקט שמייצג את המכוניות בזמן אקראי כלשהו. בנו קוד שמזיז את המכוניות בתנועה רציפה מימין לשמאל (או ההפך), ואת השחקן בהתאם לקלט ממקשי החיצים. השתמשו בטריגר כדי לבדוק אם השחקן נדרס בדרך, ובטריגר אחר כדי לבדוק אם השחקן הגיע לצד השני בהצלחה.
2. Guitar Hero - יש spawner שמתזמן רנדומלית קוביות שיורדות מהחלק העליון של המסך , הצבע של הקוביות נבחר בצורה אקראית מבין שלושה צבעים, וכאשר הקובייה מתנגשת באובייקט אחר שנמצא בתחתית המסך על השחקן ללחוץ על המקש המתאים בהתאם לצבע הקובייה. למשל מקשים z=אדום, x=כחול, ו-c =צהוב. לשינוי צבע אובייקט בזמן משחק מומלץ להיעזר בזה.
3. דו קרב: בנו משחק דו קרב יריות בין שני אובייקטי קובייה. השחקנים יזוזו משני קלטים שונים במקלדת (למשל קובייה אחת תזוז מהחצים והשנייה מהמקשים a,s,d,w) הקוביות צריכות לעמוד אחת מול השנייה כמו כן שימו לב שהכיוון של הלייזר של כל קובייה צריך להתאים למיקום של הדמות שיורה אותו. המנצח הוא מי שהצליח לפגוע שלוש פעמים בקובייה היריבה.
4. Tetris racing - השחקן הוא מכונית, והוא צריך להיזהר לא להתנגש במכוניות שמגיעות מהכיוון השני במסלול שלו. שימו לב שהשחקן זז רק על ציר ה-x.

   אם תרצו לשפר את המשחק הגדירו את התנועה של המכוניות האחרות מציר ה-z במקום מציר ה-y, והמצלמה קצת מעל לשחקן, כך המשחק ירגיש יותר ייחודי.
5. כל משחק קלאסי אחר שאפשר לממש מאובייקטים בסיסיים והתנגשויות.

שימו לב: אין צורך לממש את כל המשחק (עם כמה רמות, חיים, ניקוד, גרפיקה וכו'), אלא רק את תהליך-הליבה.

הגשה:

- העלו את המשחק שבחרתם ל itch.io.
- העלו את הקוד לגיטהאב והוסיפו קובץ README.md המסביר מה בדיוק עשיתם, כולל הפניות לקוד.

ד. סי שארפ

[אם עדיין לא תרגלתם תכנות בסי שארפ בעזרת האקראנק או קודינגיים – חזרו למטלה הקודמת ועשו זאת].

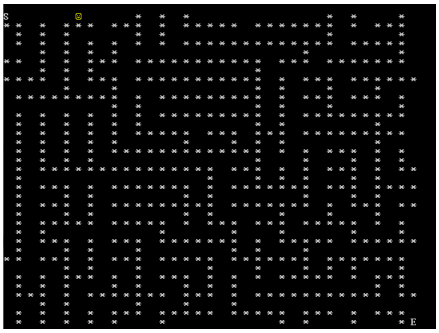
בצעו אחת מהמשימות הבאות לבחירתכם:

1. BFS- כיתבו מימוש של אלגוריתם BFS בשפת C#. בנו אותו באופן גנרי ככל האפשר. ייתכן שנשתמש בזה בהמשך למציאת מסלול במשחק.

2. איקס מיקס דריקס- כתבו משחק איקס עיגול ב-C# נסו לבנות את המשחק כך שהשחקן ישחק נגד המחשב (כמין AI).

3. איש תלוי- כתבו משחק איש תלוי ב-C# (אוצר מילים באנגלית).

4 איש במבוך- בנו משחק של איש יוצא ממבוך, השחקן ימוקם בתוך המבוך ויצטרך לצאת החוצה. התנועה שלו תהיה לפי המקשים של החצים : ימינה שמאלה למעלה ולמטה. תמונה להמחשה (אין צורך במבוך גדול) :



הגשה: קוד+תיעוד+דוגמאות הרצה בגיטהאב.