GitHub and Unity

תקציר למי שמכיר גיטהאב מקורסים קודמים / אראל

כמה דגשים לעבודה עם גיטהאב בקורס זה (וביוניטי בכלל):

א. **חשבון צוות**: אתם עובדים בצוות ולכן כדאי שיהיה לכם חשבון לצוות. בגיטהאב אפשר לעשות את זה בקלות. לא צריך לפתוח חשבון חדש עם סיסמה חדשה - צריך רק לפתוח "ארגון" חדש – Organization: לוחצים על סימן ה + בצד ימין למעלה, ובוחרים New Organization; נותנים שם מקורי לצוות שלכם; לוחצים על People ואז על Invite, ומזמינים את כל חברי הצוות להצטרף לארגון, כך שמי שנכנס לדף של הארגון יראה את כולם. מעכשיו והלאה, בכל פעם שאחד מחברי-הצוות ייצור מאגר חדש, הוא יוכל לבחור לשייך אותו לארגון החדש שיצרתם.

ב. **מאגר לכל משחק**: משחקים ביוניטי הם מאד כבדים, עם הרבה קבצים גדולים. אם תעלו את כולם לגיטהאב אתם עלולים לקבל הודעה שהחשבון שלכם נחסם. לכן כדאי מאד שכל משחק שאתם יוצרים ביוניטי, גם במטלות השבועיות וגם במטלה המתגלגלת, יהיה במאגר (repository) נפרד. כשאתם יוצרים משחק חדש ביוניטי, נוצרת לכם תיקיה המכילה כמה תת-תיקיות, כגון Assets, Packages, ProjectSettings; התיקיה הזאת צריכה להפוך למאגר בגיטהאב.

ג. **התעלמות מקבצים מיותרים**: רוב הקבצים בתיקיה של יוניטי הם קבצים זמניים שהעורך יוצר. אין טעם להעלות את כולם לגיטהאב – צריך להעלות רק את קבצי-המקור. כדי לעשות זאת, צריך לשים בתיקיה הראשית של המשחק שלכם קובץ בשם "**.gitignore**" שימו לב לשם הקובץ – שם הקובץ מתחיל בנקודה ואחריה "gitignore" בלי סיומת. בקובץ הזה צריך לכתוב את כל הקבצים והתיקיות שיש להתעלם מהם. איך יודעים ממה להתעלם? - אפשר להיעזר בקבצים ששמתי בתיקיות של הקורס, למשל כאן: https://github.com/erelsgl-at-ariel-gamedev/06-3d-terrain-ai

ד. **ניהול קבצים גדולים**: קבצי קול, תמונה ואנימציה הם בדרך-כלל גדולים ותופסים הרבה מקום על השרת של גיטהאב. ניתן להגיד לגיטהאב לשמור אותם בשרת מיוחד יעיל יותר, שנקרא LFS – Large File Storage. כדי לעשות זאת צריך לשים בתיקיה הראשית של המשחק שלכם קובץ בשם "**.gitattributes**" שימו לב לשם הקובץ – שם הקובץ מתחיל בנקודה ואחריה "gitattributes" בלי סיומת. אפשר להיעזר בקובץ ששמתי כאן: https://github.com/erelsgl-at-ariel-gamedev/06-3d-terrain-ai/blob/master/.gitattributes

ה. **קובץ רידמי**: הדבר הראשון שאנשים רואים כשהם נכנסים למאגר שלכם בגיטהאב זה הקובץ **Readme.md**. זה חלון הראווה שלכם – חשוב שהוא יהיה ברור ויפה. שימו בו קישור למשחק שלכם באתר itch, וכן הסבר על מהות המשחק ואיך משחקים בו. הוסיפו צילומי מסך לפי הצורך. מומלץ גם לשים סרטוני הדגמה של המשחק, שאפשר ליצור בעזרת תוכנות חינמיות כגון https://www.screentogif.com/ .
בנוסף, כיוון שאנחנו בקורס תיכנות, חשוב לשים בקובץ רידמי גם הסברים על הקוד: איזה רכיבים יצרתם ואיך תיכנתתם אותם. שימו קישורים מהקובץ רידמי לשורות הקוד הרלבנטיות.
לסיום, הוסיפו קישורים לכל המשאבים שנעזרתם בהם – תמונות, קבצי-קול ונכסים נוספים שהורדתם, מדריכים באינטרנט שנעזרתם בהם, וכו'.

ו. **בדיקה**: אחרי שדחפתם את המשחק לגיטהאב, ודאו שהמאגר שלכם נראה בערך כך: https://github.com/erelsgl-at-ariel-gamedev/06-3d-terrain-ai כלומר הוא מכיל רק שלוש תיקיות (Assets, Packages, ProjectSettings) ושלושה קבצים (.gitattributes, .gitignore, Readme.md). אם אתם רואים שם תיקיות נוספות, למשל Library, סימן שטעיתם בשלב כלשהו בדרך. למרבה הצער, כשקבצים גדולים נכנסים למאגר פעם אחת, הם לא יוצאים משם גם כשמוחקים אותם – הם נשארים לנצח בהסטוריה של המאגר. לכן אם טעיתם תצטרכו לפתוח מאגר חדש ולחזור על התהליך.

תזכורת - מה זה בכלל גיטהאב? / מעוז

בשביל אלו שעדיין לא יצא להם, או ששכחו בינתיים כמה מהפיצ'רים של גיטהאב, להלן סקירה מקוצרת:

גיטהאב היא מערכת אינטרנטית לניהול גרסאות- כל מסמך שאנחנו מעלים לאתר , אם העלנו אותו כמה פעמים אנחנו יכולים לברור מבין העלאות שהיו לנו (או העלאה האחרונה) איזו גרסה של המסמך אנחנו רוצים.

במה זה עוזר לנו? נניח יש לנו פרויקט גדול הבנוי ממלא מסמכים, סקריפטים וכדו', ואנחנו במקרה עשינו טעות באחד הקבצים, או שניסינו משהו חדש שלא ממש הצליח. גיטהאב מאפשרת לנו לחזור אחורה לאותה נקודת מפנה לפני שהפרויקט קיבל "תפנית שלילית" ולאתחל את הפרויקט מאותה נקודה. כמו בסרט חזרה בזמן, רק בלי התסביכים של נסיעה בזמן.

היופי בגיטהאב שהוא יודע לסנכרן בין מסמכים שונים בפרויקטים גדולים, וכך ניתן לנהל פרויקטים גדולים, לחלק תפקידים מבלי ששניים שעובדים על שני דברים שונים מאותו פרויקט יתנגשו בטעות (אלא אם אנחנו עובדים על אותו מסמך בדיוק).
לגיטהאב יש כמה תכונות שימושיות כמו למשל היכולת ליצור ענפים חדשים - אם אנחנו רוצים לקחת את הפרויקט שעשינו עד כה ולנסות לעשות לו איזשהו פיווט (pivot), כלומר לנסות לקחת אותו למקום אחר, אבל מבלי לפגוע במה שעשינו עד עכשיו. ניתן לעשות זאת ע"י יצירת ענף (branch) חדש לפרויקט, ואז נעבוד על אותו ענף, ובקלות ניתן לחזור לאותו לפרויקט הישן שעבדנו עליו ע"י החלפת ענפים.

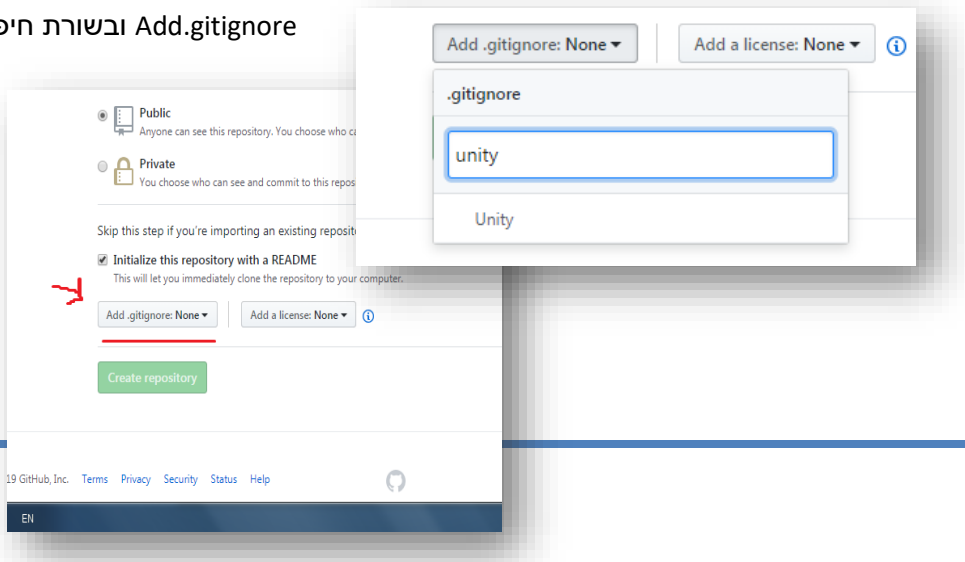
GitHub Desktop

תוכנה המאפשרת לנו לנהל את חשבון הגיטהאב שלנו משולחן העבודה. היא מקלה עלינו בעיקר בלשלוח מסמכים לאתר לא ע"י גרירה ב-update , או שימוש בחלון הפקודה של GitHub שלא נראה הכי ידידותי בהתחלה.
התוכנה מספקת ממשק משתמש נח ומובן יחסית להעלאת פרויקטים, החלפת ענפים, והחלפת מסמכים עדכניים בcommits ישנים.
למען הנוחות אנחנו נשתמש ב- GitHub Desktop, על אף שקיימות דרכים אחרות לקשר בין unity ל-GitHub משום ש:
א. קיימים הרבה מדריכים שימושיים לGitHub desktop
ב. רוב ה-assets של unity שקשורים לגיטהאב לא הכי עדכניים וגורמים לבעיות בזמן העברה של הנתונים.
ג. הכי קרוב לממשק משתמש של האתר.
אז לפני שנמשיך במדריך ,אני ממליץ למי שאין עדיין את ה-GitHub desktop להוריד אותו מהאתר:

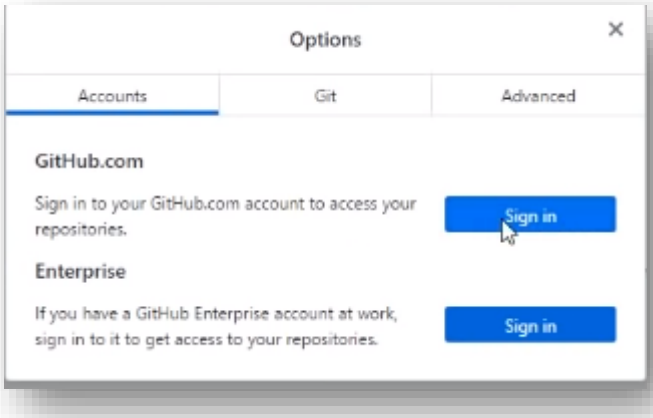
https://desktop.github.com/

שימוש בגיטהאב לפרויקטים בunity

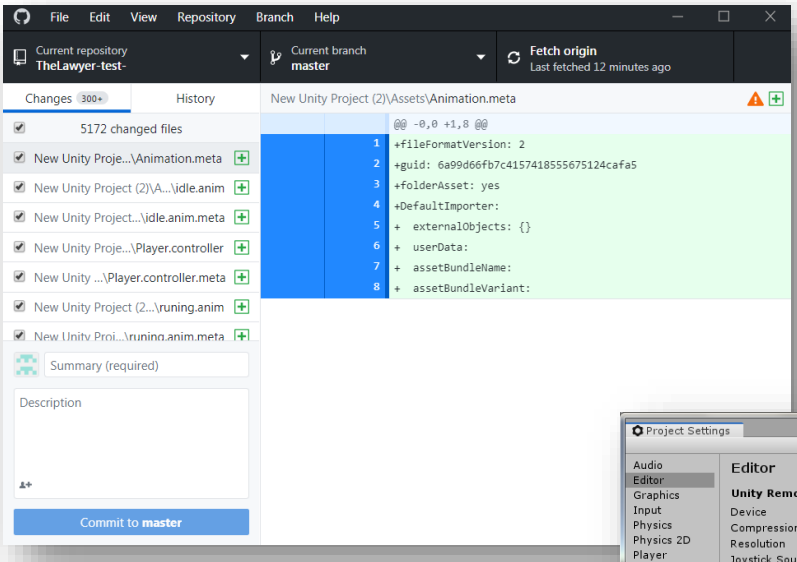
ניצור repository חדש בגיטהאב.
ניתן לו שם (אם רוצים גם תיאור), ונחליט אם נרצה שהפרויקט יהיה ציבורי או פרטי.
נלך לתחתית העמוד, נראה שיש Add.gitignore , gitignore הוא למעשה קובץ שאנחנו מגדירים מראש שנותן לנו את האפשרות לסנן בהעלאה קבצים שלא נצרכים לנו. מסתבר שלגיטהאב יש גם gitignore מיוחד בשביל unity, שמסנן קבצי מטה-דאטה שהמנוע מייצר אבל לא ממש מועילים למשהו.
נבחר ב- Add.gitignore ובשורת חיפוש נכתוב unity:

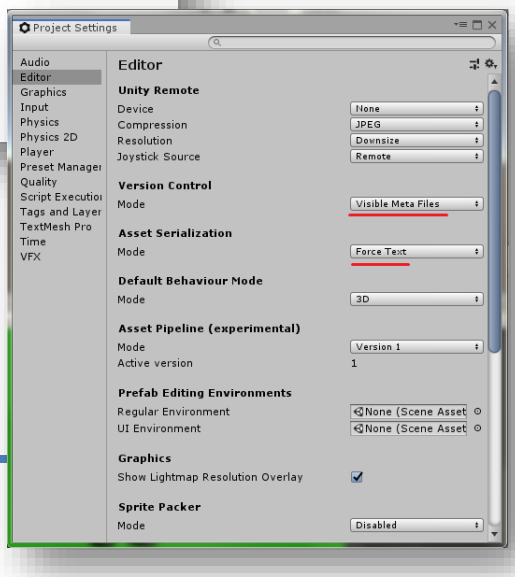


נפתח את ה GitHub Desktop . בפעם הראשונה שניכנס אליו, אם לא הגדרנו עדיין חשבון גיטהאב, נלך ל-option<-file וליד ה-GitHib.com נבחר sign in , ושם נזין את השם משתמש והסיסמא שלנו בגיטהאב כדי להתחבר:



כדי לייבא repository מהאתר נבחר באופציה clone a repository , ונכניס את ה- url של אותו repository.
נבחר היכן אנחנו רוצים שזה ישב על המחשב שלנו(באיזו תיקייה), ו-clone.
ניצור פרויקט חדש ב-unity, אם נשים לב מתחת לבחירת השם לפרויקט ניתן לבחור באיזו תיקייה במחשב נרצה לשים את המשחק שלנו. נבחר את אותה תיקייה של ה-repository שהרגע ייבאנו דרך ה-GitHub Desktop .
במידה ואנחנו רוצים לשמור פרויקט שכבר התחלנו לעבוד עליו נעתיק את התיקייה של הפרויקט unity לתיקיה של ה-repository , וב-unity hub כנראה לא יפתח לנו הפרויקט אם נלחץ עליו (כי שינינו את המיקום שלו), לכן נבחר ב-Add -> התיקייה של הפרויקט unity שהעברנו.
אם נחזור לgithub desktop נראה שהוא מראה לנו את כל השינויים שחלו בתיקייה של הrepository מאז ה- commitהאחרון שעשינו:





לפני שנעשה commit

אנחנו צריכים להגדיר כמה דברים ב-unity :
נחזור ל- unity ונבחר Edit-> project setting -> Editor וב- editor נדאג שה- version control יהיה על visible meta file.
לכל asset שאנחנו יוצרים בפרויקט, unity מייצר meta file שמכיל מידע על

אותו אובייקט שממנו הוא נוצר ומתי משתמשים בו בפרויקט, אנחנו רוצים שיהיה ניתן לראות את ה-meta files כדי שהם יבחרו ע"י ה-version control.
אם נרצה להפוך את הקבצים של unity גם ליותר קראים נבחר Asset serialization -> force text:

אחרי שעשינו את זה נוכל לחזור לGitHub desktop ולעשות commit.
ה commit שעשינו מעודכן רק ב -git המקומי שלנו, אבל לא באתר. כדי לעלות אותו לאתר נצטרך לעשות גם push to origin .
עכשיו כל פעם שנעשה איזשהו שינוי במשחק הוא ישר יתעדכן בתיקייה של ה-repository ונוכל לעשות לו commit ו-push .
במידה ועבדנו על הפרויקט ממחשב אחר, או שאנחנו עובדים עם כמה אנשים ואנחנו רוצים לעדכן את התיקייה במחשב שלנו עם ה-commit האחרון שהתבצע נלחץ על Fetch origin, שנמצא בדיוק היכן שהכפתור של הpush, ואז נלחץ pull (או ctrl+shift+P שעושה pulling אוטומטית).