חלליות בשני ממדים – חלק שני

עד עכשיו התעסקנו בבניית שלד למשחק שלנו- תנועה של הדמיות, לוגיקת משחק ועיצוב קל ,כדי לדמות איך המשחק שלנו יתנהל. עכשיו נתעסק בחלק ה"אומנותי" יותר של המשחק- עיצוב דמויות ורקעים, UI ,תפריט ראשי ועוד.

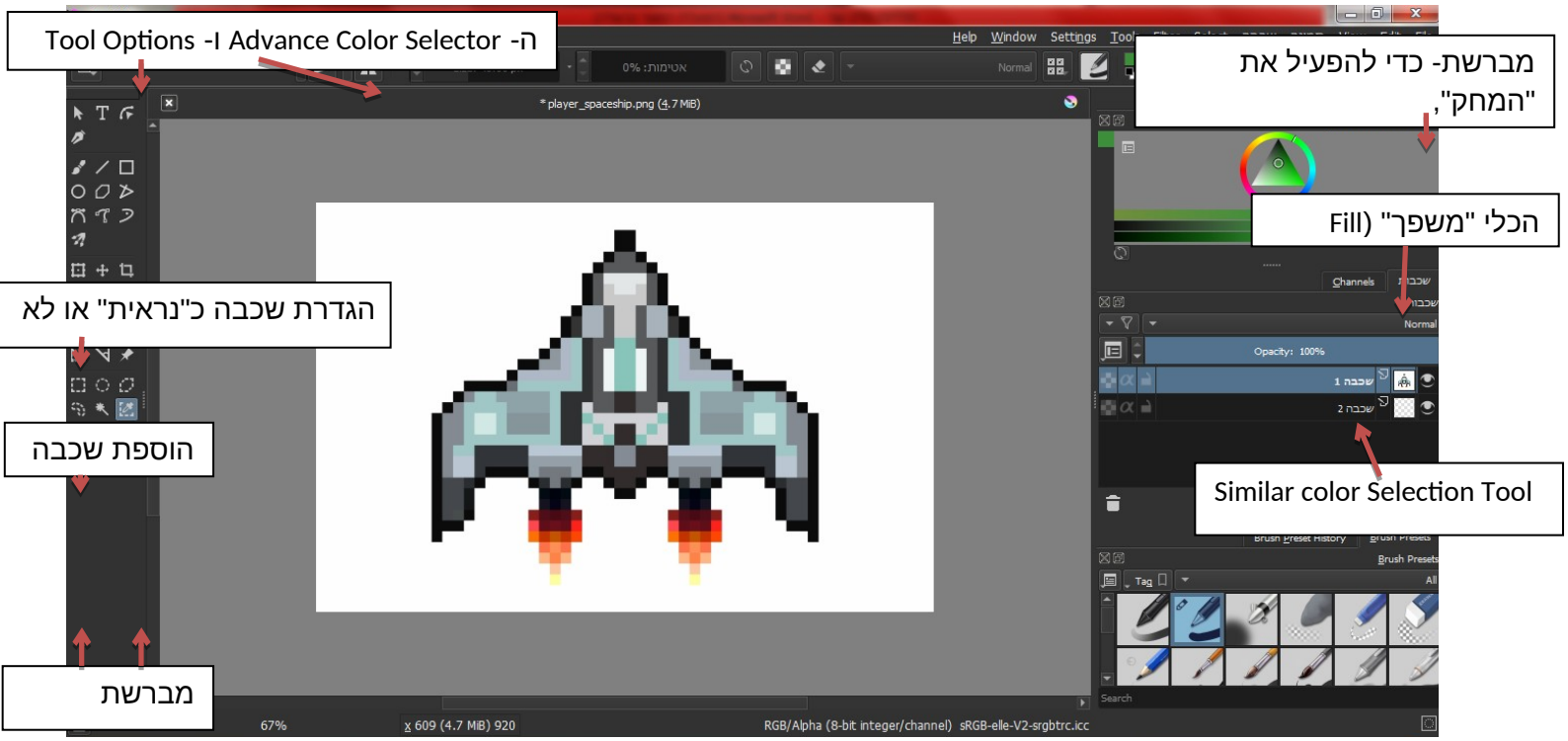
מכאן נוכל לקח את המשחק שלנו לשני כיוונים- או שנמשיך עם אותו כיוון שהתחלנו אתו וניישם משחק תלת-ממדי, שהיתרונות בו ברורים-ממשיך באותו הקוד שהשתמשנו בו קודם(אין שינויים קטנים),מרהיב יותר וראליסטי יותר, אך הוא ישקול בסופו של דבר הרבה יותר. או שנשנה כיוון ונבנה את המשחק שלנו דו-ממדי, שהוא הרבה יותר קל ליישום, שוקל פחות, ומתאים ליותר פלטפורמות.

הבחירה היא בידיים שלכם ושתי הדרכים לגיטימיות, גם ההבדלים מבחינת התהליך לא גדולים במיוחד, אך לשלב הזה עדיף שנתמקד שבמשהו שיותר קל ליישם ולכן אנחנו ממשיכים כדו-ממד, אבל שוב מי שמעדיף ליישם את המשחק כתלת-ממד מומלץ לו להמשיך אתנו כי רוב החומר דיי זהה.

מעבר מ3D ל2D-

בשביל להמיר את הדמויות שלנו לדו-ממד נצטרך assets דו-ממדיים בשביל המשחק שלנו. מומלץ ביותר לחפש אם קיימים assets שעונים לנו על רוב הדרישות בחנות של (unity asset store) unity, כמעט בטוח שנמצא שם ערכה שכוללת **הכל**, החל מדמויות וכלה בעזרים כמו אודיו או אנימציות במיוחד למשחק שלנו.
חשוב לציין שרוב ה asset בחנות בתשלום, אך יש מגוון ענק של asset בחינם שניתן לייבא למשחק ,ואפשר למיין לפי מחירים.
במידה ולא מצאנו ,או שאנחנו רוצים לייצור בעצמנו אל דאגה גם בזה נטפל. במשחק נצטרך כמה דברים שיחליפו את האובייקטים הפרימיטיביים שהשתמשנו בהם עד כה : 1) תמונת רקע- התמונה לא חייבת להיות בפורמט ספציפי, כל זמן ש-unity יכול לקרוא אותו, לרוב עדיף להשתמש בפורמט png כי הוא שומר על איכות התמונה המקורית. קל למצוא תמונות שמדמות חלל חיצון בגוגל. 2) דמויות או אובייקטים- התמונות של הדמויות\אובייקטים צריכות להיות בפורמט png ספציפית, **ובלי רקע**, לרוב קשה למצוא תמונות כאלה, לכן נצטרך ליצור כאלה בעצמנו ע"י תוכנת גרפיקה כלשהי, המומלצות הן Photoshop או , Krita אומנם לפוטושופ יש הרבה יותר מדריכים שימושיים והיא עם תמיכה טכנית ,אך אישית אני ממליץ על krita משתי סיבות עיקריות : א) היא חינמית- היא תוכנת open source שרצה על כמה מערכות הפעלה(linux בניהן) ואין צורך ברישיון מיוחד כדי לעבוד איתה. ב) יש לה פיצ'רים במיוחד לאנימציה ועיצוב גרפי שמתאימים לבניית משחקים אינדיים (עצמאים). לענייננו: איך נפריד בין התמונה לרקע שלה? ברמת העיקרון אם אנחנו בונים בעצמנו את הדמויות אין לנו צורך בפורמט ספציפי (GIF,IPJE,PNG) ובלבד שהרקע יהיה בצבע שונה מהאובייקט שישמש אותנו.
תמונה לדוגמא:

נוריד את התמונה למחשב ונפתח אותה באמצעות קריטה. לקריטה יש כמה כלים שיכולים לעזור לנו להפריד בין הרקע לתמונה עצמה. ראשית נצטרך לדאוג שתהיה לנו שכבה חדשה מתחת לתמונה ,לכן ניצור שכבה חדשה ונגרור אותה שתהיה מתחת לשכבה של התמונה ונגדיר אותה כ"לא נראית" (אייקון שדומה למין עין בצד של האייקון של השכבה). נחזור לשכבה של התמונה ,נבחר בכלי similar color selection שמסמן בשכבה את אותם המקומות שיש להם את אותו הצבע. צריך לדאוג שה- fuzziness שלו יהיה נמוך (בין 1 ל-5),כך הוא יבדיל כמה שיותר בין גוונים שונים, אפשר לעשות את זה ב- tool option. נלחץ על הרקע עם כלי, הוא אמור לסמן לנו רק את הרקע(זה נראה כמו מלא קווים מסביב לדמות) . נמחק את מה שסימנו, או ע"י delete או ע"י מברשת "מחק" ונמחק פשוט אקטיבית את האזור המסומן. כדי לבדוק שלא התווספו לנו "שאריות מהרקע" נסמן את השכבה של התמונה כ"לא נראית", ונעבור לשכבה מתחת ונסמן אותה כ-"נראית", נבחר בכלי משפך ונתן לו צבע בולט ששונה לגמרי מהצבע של הדמות (לרוב ירוק כמו מסך ירוק בסרטים הוליוודיים). ונחזיר את התמונה להיות "נראית", במידה ויש שאריות פשוט נמחק אותם ע"י המברשת "מחק", אם אין שאריות פשוט נסמן את השכבה התחתונה כ"לא נראית" ונשמור את התמונה כקובץ png ב- save as -> File (או ctrl+shift+s) ובחלון שנפתח נבחר ב- save as type כ- png.
זה אמור לשמור לנו את התמונה ללא הרקע שלה. באיור למטה יש לנו תצוגה של הכלים של קריטה שהשתמשנו בהם.



בשביל לעלות ל unity (בין את התמונת רקע ובין את התמונה של האובייקטים) פשוט נגרור את האובייקט למנוע הגרפי והוא יופיע בחלון הפרויקט תחת השם המקורי שנתנו לתמונה, וב-inspector שלו נשנה את ה Texture Type ל- Sprite(2D and UI) אח"כ נוכל לגרור את האובייקט לחלון הסצנה ולראות אותו בחלון המשחק גם כן.
יכול להיות שלמרות שהגדרנו את התמונה כמו שצריך, וכן ניתן לגרור אותה לחלון הסצנה אך עדיין אנחנו לא רואים את האובייקט בחלון כמו שרצינו. כנראה שהסיבה לכך היא משום שהתמונה לא נמצאת מעל לשכבת הרקע.
אם נסתכל על ה inspector של הרקע נראה שיש שם רכיב שקוראים לו additional setting ולו שני אלמנטים: Sorting Layer ו- Order in Layer, הראשון אחראי על איזו שכבה יופיע בה הרקע, מומלץ ליצור שכבה חדשה במיוחד לרקע ע"י Add Sorting Layer->. השני משני האלמנטים הוא המיקום של האובייקט יחסית לשכבה בה הוא נמצא כאשר 0 הוא הכי

נמוך 1 מעליו וכן הלאה. נגדיר את הרקע כמיקום 0 ואת האובייקט כמיקום 1 או יותר (תלוי כמה אובייקטים הגדרנו) , כך נוכל לראות אותו גם בחלון המשחק.

המרת הדמויות ל D2

אחרי שיש לנו כבר תמונות לייצוג השחקן האויבים והלייזר אנחנו רוצים להחליף בין האובייקטים הפרימיטיביים שמשמשים את הדמויות, לאובייקטים הייעודיים להם. בשביל ליצור את player פשוט נגדיר אובייקט player חדש ונגרור אליו את הסקריפט של player. אם נמחק את האובייקט player הישן שהשתמשנו בו , נגרור את האובייקטים הרלוונטיים ל inspector של השחקן החדש (במקרה שלנו לייזר),נציין בתג שלו שהוא player ונתחיל לשחק, נראה שהשחקן אומנם יוצר לייזר תלת-ממדי כמו מקודם שמצליח לפגוע 'באויב' הפרימיטיבי שעוד לא החלפנו, אך אם השחקן פוגע בגופו באויב לא מתרחשת "התנגשות" בין האובייקטים, למה זה? זה משום שהשחקן שלנו היה מוגדר כאובייקט תלת-ממדי ועכשיו הוא דו-ממד, לדו-ממד ולתלת-ממד יש
רכיבים שונים להתנגשויות, כפי שנראה עוד מעט באובייקטים הפרפאביים(מלשון prefab) שלנו.

נבחר ב prefab- "אויב" ובחלון ה inspector נסיר ממנו כל אלמנט תלת-ממדי שמגדיר אותו: הקובייה,,mesh renderer box collider ו- rigid body כדי להסיר אותו פשוט נלחץ מקש ימני מעל אותו רכיב-> Remove Component. לאחר שהסרנו את הרכיבים הנ"ל אנחנו רוצים שיהיה ניתן לראות את האובייקטים, לכן נצטרך להוסיף להם רכיב sprite שבו נאחסן את התמונה שמייצגת אותו. לשם כך נבחר ב- Add Component ובשורת החיפוש נכתוב sprite renderer ונוסיף. נגרור את התמונה של האויב לתוך התפריט של ה sprite והתמונה תופיע מולנו, חשוב להגדיר את האובייקט בשכבה המתאימה לו. אם נריץ את המשחק התמונה של האויב שהגדרנו מופיעה מולנו, אך עכשיו אפילו הלייזר לא פוגע בה. כדי לפתור את זה נוסיף לאובייקט עוד שני רכיבים: rigidbody2D ו- box collider2D(ב box collider2D נסמן גם את is Trigger). גם לשחקן נוסיף את הרכיבים האלה וגם לו נסמן את is Trigger

עדיין לא סיימנו! עכשיו נצטרך לשנות את הסקריפט בהתאם. בסקריפט נשנה את שם המתודה `OnTriggerEnter(Collid er other)` ל- `OnTriggerEnter2D(Collider2D other)`. וזהו, אם נריץ עכשיו נראה שהאויב יכול להתנגש בשחקן, אבל משום שלא שינינו את הלייזר להיות דו-ממדי הוא עדיין לא יושפע ממנו. יכול להיות שהתנגשות של האויב בשחקן קורה לפני הזמן, כלומר עוד לפני שהשחקן ממש נוגע באויב האויב מושמד. כנראה שהבעיה שגבולות ה collider מוגדרות מעבר לגבולות התמונה, כדי לשנות את גבולות ה colliders של השחקן או האויב צריך ללחוץ על Edit Collider ב inspector ולשנות אותו לרצוי לנו.

שימו לב! המנוע הפיסיקלי ל-2 ממדים הוא נפרד לגמרי מהמנוע הפיסיקלי ל-3 ממדים. לכן, אם עצם אחד הוא "מתנגש דו-ממדי" למשל BoxCollider2D, והעצם השני הוא "מתנגש תלת-ממדי" למשל BoxCollider, הם לא יתנגשו במשחק! באותו אופן, אם יש שני מתנגשים דו-ממדיים, אבל באחד מהם מוגדרת פונקציית הזיהוי להתנגשויות תלת-ממדיות (OnTriggerEnter), העצמים יתנגשו במשחק, אבל הפונקציה לא תופעל! דו-ממד ותלת-ממד הם שני עולמות נפרדים לגמרי ביוניטי.

עכשיו לאחר שראינו איך לשנות את ה-prefab "אויב" יהיה לנו קל לשנות את הלייזר בהתאם. ניצור אובייקט לייזר חדש עם התמונה של הלייזר, נמחק את האובייקט הישן של הלייזר, נוסיף לו box collider2D, sprite renderer ו- rigidbody2D, נסמן אותו כ- is trigger,נוודא שלא מסומן ה gravity, ונשנה את הקוד שלו בהתאם. גם כאן- אם האובייקט מתנגש מוקדם מידי נערוך את גבולות ה collider שלו, ונבדוק כמובן שהמשחק רץ כמו שצריך.

Collider-

לא תמיד נשתמש ב boxcollider2D לפעמים השימוש במסגרת ריבועית לא מתאים לגבולות הדמות שלנו, כלומר אם הדמות שלנו עגולה או עם עקומות דווקא נעדיף להשתמש ב collider מעוגל כמו polygonCollider או circlecollider.
להלן רשימה עם הסברים ל colliders הדו-ממדיים שניתן להשתמש בהם:
BoxCollider2D- יוצר גבול מלבני ל collider.
CircleCollider2D- כמו ה box רק שגבולות ה collider הן עגולות.
EdgeCollider2D- נותן אפשרות לסמן את גבולות ה collider בצורה אינטראקטיבית ע"י הוספת קדקודים והזזתם.
PolygonCollider2D- יוצר את גבולות ה collider בצורה גסה לפי צורת האובייקט.