המנוע הפיזיקלי של unity

על מנת שתהיה התנהגות פיזיקלית משכנעת, אובייקט במשחק צריך להאיץ בצורה נכונה ולהיות מושפע מהתנגשויות, גרביטציה וכוחות נוספים. המנוע הפיזיקלי של unity ,המגיע עם התקנת התוכנה, מספק רכיבים המשמשים להדמיה הפיזיקלית בשבילנו.
בעזרת הגדרת כמה פרמטרים פשוטים, נוכל ליצור אובייקטים המתנהגים בצורה פסיבית בדרך ראליסטית ,כלומר הם יזוזו כתוצאה מהתנגשויות ויפלו אך לא ינועו בכוחות עצמם.
באמצעות שליטה על הפיזיקליות ע"י סקריפטים, נוכל לתת לאובייקטים דינמיקה של רכבים, מכונות או אפילו חתיכות בד.
במהלך הפרק הקרוב נעשה סקירה קצרה על היכולות של המנוע הפיזיקלי של unity. נגדיר התנהגויות פיזיקליות מחיי היום יום ונראה כיצד נוכל לבטא אותם באמצעות כמה שורות קוד פשוטות או בהגדרת כמה פרמטרים באינספקטור.
כמו כן נפגוש כמה סוגי משחקים (חלקם אף מוכרים), וכיצד המנוע הפיזיקלי בא לידי ביטוי באותם משחקים.

המערכת הפיזיקלית של unity-

כשאנחנו חושבים על משחקים מבוססי פיזיקה לרוב אנחנו חושבים על התנהגויות בין גופים קשיחים שונים- חישוב וביצוע של אינטראקציות ראליסטיות בין קבוצות אובייקטים.
שלושת ההיבטים המרכזיים של מערכת פיזיקלית הם:
אינטגרציה- איך הפיזיקה מתאימה לעולם המשחק.
Collusion detection- אילו אובייקטים חופפים ואיך המערכת מבחינה בכך.
collusion reaction- מטפל בתגובות ראליסטיות בהתאם ל-collusion detection, למשל כדור גומי שמתנגש בקיר וקופץ.
כפי שכבר יצא לנו לראות בשיעורים הקודמים, השימוש ברכיבים פיזיקאליים ב-unity הוא אופציונלי. אם נרצה להוסיף לאובייקט איזושהי התנהגות פיזיקאלית נצטרך להוסיף לו את הרכיבים המתאימים לכך(Rigidbody, collider...).
למעשה לunity יש שתי מערכות פיזיקליות שונות: אחת המשמשת למודלים בתלת-ממד ,ואחת המשמשת למודלים בדו-ממד. הרעיון המרכזי די זהה בין שתי המערכות השונות (למעט הממד הנוסף שיש ב3D), אך הם מיושמים ע"י רכיבים שונים. כבר יצא לנו לראות בשיעורים הקודמים את ההבדל בין הרכיבים למשל עבור תלת ממד יש לנו את ה-Rigidbody ואילו בדו ממד נשתמש ברכיב Rigidbody2D בשביל לדמות גוף קשיח.

קצת רקע על המערכת הפיזיקלית של unity-

עד לאחרונה השימוש במערכת הפיזיקאלית של unity נעשה באמצעות הספרייה physx, ספריית open source מתפתח שנותנת מענה פיזיקלי להרבה מאוד פלטפורמות שונות.
השימוש בספרייה היה לצורך משחקי תלת-ממד, עבור דו-ממד יש את הספרייה Box2D.
היתרונות הבולטים של השימוש בספריות אלו היה לא רק בשל היותם open source או היכולת להתאים אותן להרבה פלטפורמות, כי אם ניהול הזיכרון המעולה והיכולות הmultithread -יות שלהן.
נכון ל2019 unity הכריזה בוועידה לפיתוח משחקים (GDC) כי היא תיתן את האפשרות לבחור בין שתי פלטפורמות לסימלוץ התנהגות פיזיקלית- "unity physics" ו- "Havok" המבוססות על המערכות הפיסיקליות הישנות.

-Player setting

אם לא יצא לנו להכיר עדיין, Unity Project Setting הוא החלון בו ניתן לשנוי הגדרות מתקדמות של המשחק - רזולוציות, איכות סאונד, הגדרת קלטים, פיזיקה בדו ובתלת ממד ועוד.
ההגדרות כאן הן דיפולטיוביות, כלומר אם נשנה אותן נשנה לאורך כל המשחק.
נפתח את החלון ע"י: Edit -> Project setting.
אנחנו נתמקד על כמה שדות מפתח מתוך המאפיינים הפיזיקאליים:
Gravity- לפני שנמשיך קצת רקע על גרביטציה:
בהגדרה הגרביטציה (או כוח הכבידה), היא אחת הכוחות הבסיסים בטבע.
כוח משיכה פועל בין גופים בהתאם למכפלת המסות שלהם ומרחקם אחד מהשני (המונח "כוח הכובד" מתייחס בדרך כלל לכוח בו כדור הארץ מושך אליו עצמים על פני שטחו) [ויקיפדיה].
אומנם זה לא מורגש ,אך הכוח משפיע גם על משיכה בין גופים קטנים כגדולים, כלומר קיימת משיכה פיזיקלית בין בני אדם, אך משום שמסותיהם של האנשים לעומת המסה של כדור הארץ זניחה אז לא מורגש אותו כוח. על אותו עיקרון, כדור הארץ נמשך לגופים קטנים, כלומר כדור הארץ נמשך אלינו מאותו כוח, אך לעומת המסה שלו אנחנו בקושי משפיעים על המשיכה, לעומת זאת כדור הארץ והירח האפקט ברור לעין- הירח מקיף את כדור הארץ בגלל אותו כוח, ובכדור הארץ קיימות תופעות כמו גאות ושפל הנגרמות מהכוח הכבידה של הירח.
כאשר אנחנו מדברים על כוח המשיכה או כוח g (על כדור הארץ) של כוכב, אנחנו בעצם מדברים על כוח תאוצה שקול שהוא שילוב של המשיכה מהמסה של כדור הארץ והכוח הצנטריפוגלי הנוצר מסיבוב הכוכב.
תאוצה זו נמדדת ביחידות של מטר לשנייה בריבוע (m/s^2).
קרוב לפני שטח כדור הארץ תאוצת הכבידה היא בערך $9.81\ m/s^2$, למעט אזורים כמו מפרץ הדסון בקנדה (אם אתם מתקשים לעמוד בדיאטה כדי שתשקלו פשוט לעבור למפרץ הדסון ,שם תשקלו פחות) .
עד כאן הרקע וחזרה לעניינו: אם נסתכל על אחד מההגדרות שקשורות בפיזיקה של המשחק(Physics או Physics2D) נראה שם שדה שקוראים לו Gravity, וכשמו כן הוא- מנסה לדמות את הגרביטציה(או, יותר נכון, כוח המשיכה) על כדור הארץ. הכוח מסומן על פני ציר ה-y כ 9.81- (הוא מתבטא כנפילה "חופשית"). כמובן שהכוח הוא אופציונלי, למשל אם נרצה ליצור משחק שמתרחש במאדים או סתם סימולציה של הכוכב נוכל לשנות את אותו כוח ל -3.711 .
Default Material- ניתן להגדיר את סוג חומר שיתווסף לכל אובייקט חדש ששמנו לו Rigidbody כלשהו.
כדי ליצור חומר חדש נבחר Asset -> Create -> Physical Material בחומרים נוכל להגדיר- רמת חיכוך(האם הוא מחליק על המשטח כמו קוביית קרח) ורמת קפיצות של החומר, למשל עבור אובייקט סטטי כמו אבן נעדיף להגדיר רמת קפיצות נמוכה, ועבור אובייקטים עשויים גומי נעדיף רמת קפיצות גבוהה.
Layer Collision Matrix- שימו לב שבאותו החלון ,בלשונית של אחד ה-physics, למטה יש לנו כמין פירמידה כזאת של שכבות.
הפירמידה מסמלות אילו שכבות משפעות על האובייקט.

נסו זאת בעצמכם:
<צריך להגדיר להוריד קובץ עם אובייקט שיש לו חומר ומשטח משופע כלשהו. את הקובץ צריך לעלות לתיקיה של ה"קוד" >

2D Physics –

כבר יצא לנו להכיר חלק מהרכיבים, אך בכל זאת נעשה סקירה כללית ליישור קו:

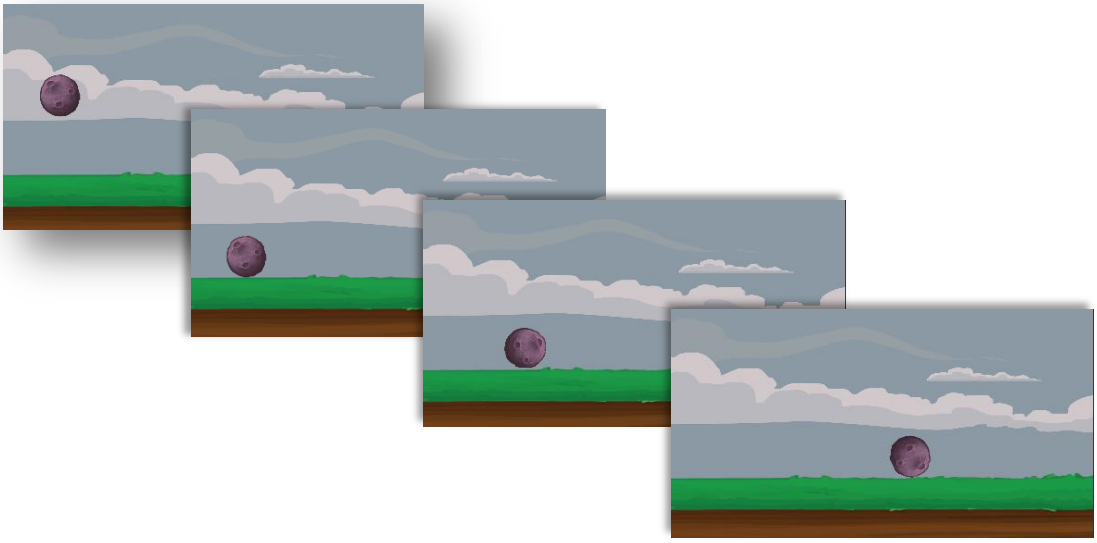
Transform- הרכיב הבסיסי ביותר של אובייקטים מסוג monobehaviour. הוא מגיע אוטומטית ביצירת אובייקט משחק חדש.
אחראי לפוזיציה, רוטציה(סיבוביות) וגודל האובייקט.

Rigidbody2D – רכיב שמגדיר את האובייקט משחק כגוף קשיח, ובכך מאפשר לו התנהגות פיזיקלית מסוימת.
הרכיב לא מגיע עם האובייקט ויש להוסיף אותו ב-Add component ולחפש בשורת החיפוש Rigidbody2D.
כברירת מחדל האובייקט מסומן עם גרביטציה.
נתמקד על כמה שדות עיקריים ברכיב:
* *Body type*- בunity קיימים שלושה סוגים של גופים קשיחים:
1. Dynamic- הנפוץ מבין השלושה. גוף שיש לו מיקום ומהירות, ופועלים עליו כוחות פיסיקליים כגון כבידה.
2. Kinematic- גוף שאמור לזוז אך לא מושפע מכוחות אחרים. יכול להתנגש רק באובייקטים שמוגדרים כדינאמיים.
3. Static- גוף שאמור להישאר במקום אחד כמו קיר. מתאים לייצוג או אובייקטים הקשורים לרקע. לא מגיב לגרביטציה, לא אמור לשנות את מיקומו בצורה ישירה, ואם לא מגדירים לו אחרת, לא אמור להתנגש באובייקטים שאינם דינאמיים.
**Material*- מתאר את החיכוך של האובייקט ואת רמת הקפיציות שלו בזמן התנגשות ,כפי שתיארנו למעלה.
**Drag*- הנטייה של אובייקט להאט כתוצאה מחיכוך עם האוויר או נוזל כלשהו שסובב לו. ישנם שני סוגים של drags:
1. Angular Drag - מייצג את היכולת לסובב את האובייקט. ככל שהוא יותר גבוה ככה קשה יותר לסובב אותו.
2. Linear Drag- משפיע על מיקום האובייקט, ככל שהוא יותר גבוה הוא מתאר אובייקט שקשה יותר להזיז אותו.
**Mass*- מייצג את המסה של האובייקט. ככל שהמסה גדולה יותר ככה קשה לאובייקט לזוז- הוא מנסה להיצמד יותר לקרקע או לכוח שמושך אותו כלפי מטה.
הערה: אין זה אומר שהוא נופל מהר יותר מאובייקט עם מסה פחותה (מופעל עליהם אותו כוחות).

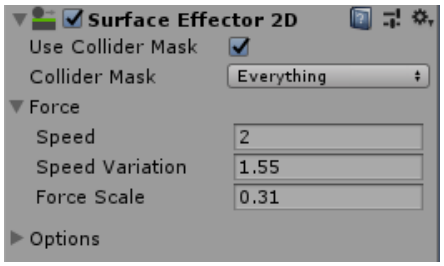
Collider2D – רכיב שמגדיר את קווי המתאר של האובייקט לצורך התנגשות, כלומר הוא מסמן את הגבולות שבהם האובייקט ירגיש את ההתנגשות. אם יש לאובייקט Collider ניתן להגדיר לו לוגיקה ספציפית בזמן התנגשות עם אובייקט אחר ע"י מתודות ייעודיות של הרכיב.
הערה: לכל אובייקט יכולים להיות כמה colliders – יכול להיות שהאובייקט בנוי מכמה תתי-אובייקטים שלכל אחד יש הגדרה אחרת להתנגשות. זאת גם אחת הסיבות שיש לנו material ב-collider על אף שכבר יש לנו ב-rigidbody, כדי שנוכל לתת לכל collider חומר אחר אם נרצה.
IsTrigger - לפעמים נרצה לדמות התנגשות עם אובייקט שאינו דינאמי לכן נשתמש בטריגרים שידמו לנו כמין שדה בלתי נראה שמעורר התנהגות פיזיקלית. כאשר אנחנו מפעילים את הפונקציות הקשורות בטריגר אנחנו בעצם מגדירים לוגיקה מבלי להתייחס לתוצאות ההתנגשות (אם יש כאלו, יכול להיות שאובייקט שאינו דינאמי התנגש בטריגר ועדיין יופעל הטריגר).

Effectors2D- רכיב שעדיין לא יצא לנו לדבר עליו. מטרת הרכיב היא לכוון את הכוחות שמופעלים כאשר אובייקט מתנגש באובייקט עם Effector.
סוגי האפקטורים:
1. *Surface Effector*- מיישם כוח לאורך פני שטח הקוליידר. כאשר אובייקט מתנגש באובייקט עם אפקטור זה, האובייקט המתנגש יתחיל לנוע לכיוון מסוים . בד"כ את הרכיב נצמיד לאובייקטים שאמורים לייצג משטח או קרקע, כך שברגע שאובייקט פגע במשטח הוא יתחיל לנוע עליו, למשל קרקע משופעת וכדו.
לדוגמא: נפיל כדור דו ממדי על משטח עם surface effector, הכדור ינוע לכיוון מסוים בהתאם לערך התנועה.

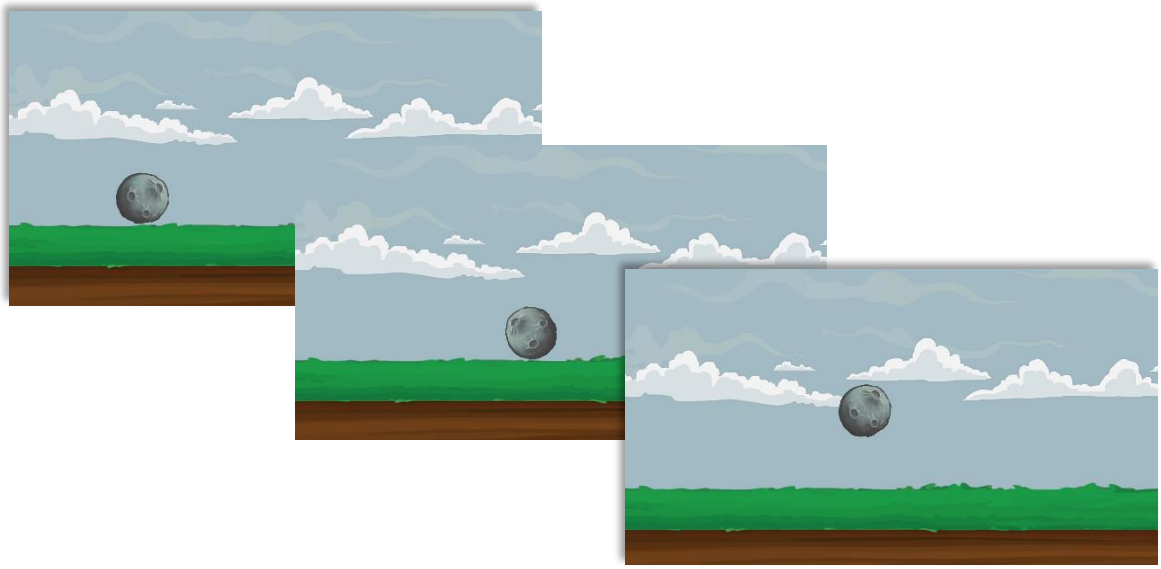
השדות של האובייקט:
Speed– נוכל להגדיר את המהירות והכיוון של האובייקט המתנגש. מספר חיובי האובייקט ינוע ימינה (הכיוון החיובי של ציר ה-x) מספר שלילי שמאלה, וככל שהערך יותר גדול(בערך מוחלט) הוא ינוע מהר יותר.
force scale- כמה זמן ייקח לאובייקט מרגע ההתנגשות להאיץ למהירות המרבית. ככל שהערך נמוך יותר ייקח לאובייקט יותר זמן להאיץ(ערכים בין 0 ל1).
speed variation – מספר רנדומלי בין 0 ל-מספר שהוזן. הערך מוסף למהירות שהגדרנו ב-speed. עבור ערך שלילי נוספים כוחות נגדיים לכוח ששמנו ב-speed.
collider mask- מגדיר באילו שכבות ישפיע האפקט.



התמונות לעיל ממחישות שימוש של האפקטור. הוספנו למשטח את האפקטור surface effector 2D ו- collider2D, בקוליידר של המשטח גם הגדרנו used by effector, כדי שנוכל להשתמש באפקטור,ולאסטרואיד הוספנו גוף קשיח וקוליידר. כאשר הכדור נוחת על המשטח בהרצת המשחק הוא "מתגלגל" ימינה במהירות.
באפקטור לצורך הדוגמא הגדרנו כך :



2. *Area Effector* – מיישם כוח במרחב, או אזור במרחב שבו מתרחש שינוי בתנועה הפיזית.
למשל נניח הכדור מהדוגמא הקודמת ממשיך לנוע על המשטח עם ה-surface effector, עד שהוא נכנס לאזור שמפעיל עליו כוח וגורם לו לקפוץ.
נמשיך מהדוגמא שעשינו למעלה- ניצור אובייקט ריק חדש ונוסיף לו colliderBox2D ובקוליידר נגדיר use by effector ו- Is Trigger כדי שנוכל להשתמש באפקטור area. נוסיף את area effector 2D ונסמן את use collider mask.
נשנה את הגודל של הקוליידר והמיקום שלו להיכן שנרצה שבו יופעל האפקטור.
בעזרת השדה force magnitude נוכל להגדיר את עוצמת הכוח שיופעל על האובייקט שיכנס למתחם של האפקטור, ובעזרת force angle את הזווית של הכוח.
דוגמאת הרצה:



האסטרואיד ממשיך לנוע לאותו כיוון עד שהוא נכנס לקוליידר של האובייקט הריק שעשינו, ואז בגלל הareaEffector מופעל על האסטוראיד כוח שגורם לו לקפוץ בזווית.