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.
Collision detection- אילו אובייקטים חופפים ואיך המערכת מבחינה בכך.
collision reaction- מטפל בתגובות ראליסטיות בהתאם ל-collision 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].
גופים קטנים מופשעים ומשפיעים על גופים אחרים, לדוגמא בני אדם הם בעלי מאסות קטנות ולכן ההשפעה תהיה מאד קטנה על הגופים האחרים עד כדי כך שההפשעה זניחה. לעומת זאת, בנושא כדור הארץ והירח האפקט ברור לעין- הירח מקיף את כדור הארץ בגלל השפעת כוח הכבידה. בכדור הארץ קיימות תופעות כמו גאות ושפל הנגרמות מהשפעת כוח 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(כח תנועה סיבובית) הנוצר מסיבוב הכוכב סביב עצמו.
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.

– 2D Physics Components

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

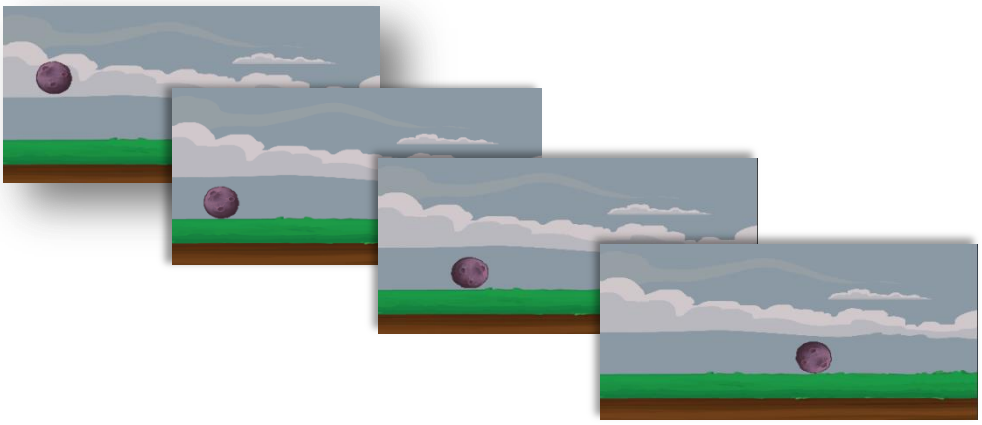
-Transform הרכיב הבסיסי ביותר של אובייקטים מסוג monobehaviour. הוא מגיע אוטומטית ביצירת אובייקט משחק חדש.
אחראי לפוזיציה, רוטציה(סיבוביות) וגודל האובייקט.

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).

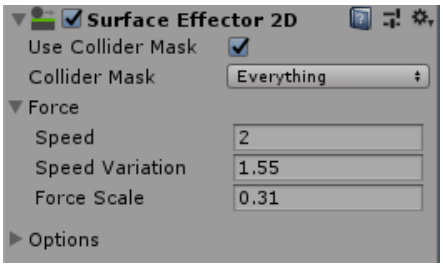
Collider2D – רכיב שמגדיר את קווי המתאר של האובייקט לצורך התנגשות, כלומר הוא מסמן את הגבולות שבהם האובייקט ירגיש את ההתנגשות. אם יש לאובייקט Collider ניתן להגדיר לו לוגיקה ספציפית בזמן התנגשות עם אובייקט אחר ע"י מתודות ייעודיות של הרכיב.
הערה: לכל אובייקט יכולים להיות כמה colliders – יכול להיות שהאובייקט 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.

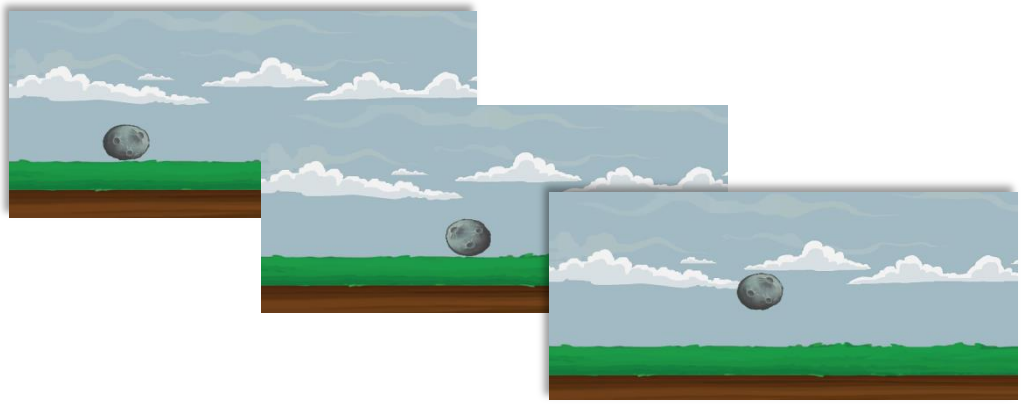
collider mask- מגדיר באילו שכבות ישפיע האפקט.



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 :

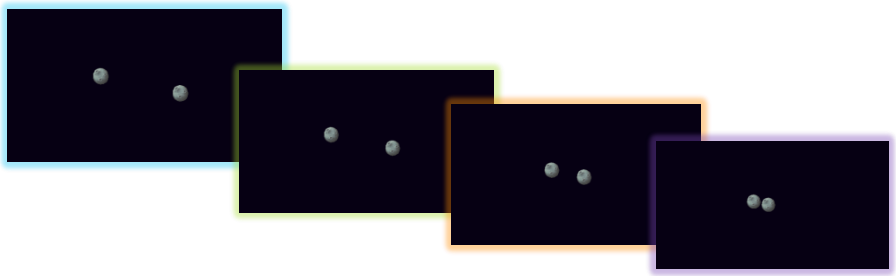


2. *Area Effector* – מיישם כוח במרחב, או אזור במרחב שבו מתרחש שינוי בתנועה הפיזית.
למשל נניח הכדור מהדוגמא הקודמת ממשיך לנוע על המשטח עם ה-surface effector, עד שהוא נכנס לאזור שמפעיל עליו כוח וגורם לו לקפוץ.
נמשיך מהדוגמא שעשינו למעלה- ניצור אובייקט ריק חדש ונוסיף לו colliderBox2D ובקוליידר נגדיר use 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:



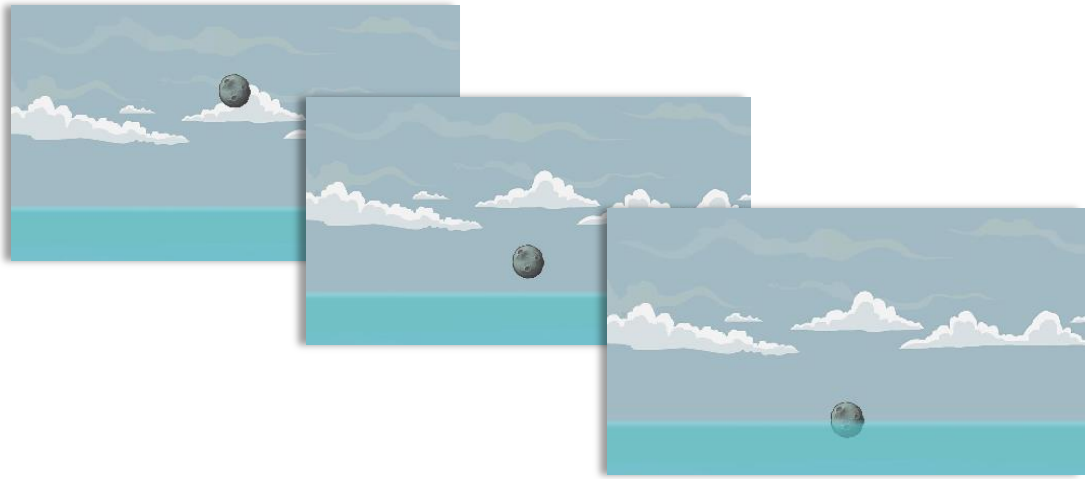
האסטרואיד ממשיך לנוע לאותו כיוון עד שהוא נכנס לקוליידר של האובייקט הריק שעשינו, ואז בגלל הareaEffector מופעל על האסטוראיד כוח שגורם לו לקפוץ בזווית.

3. *Point Effector*- מגדיר משיכה או דחיה של אובייקטים אחד לשני.
כמין מגנט שבנוי על האובייקט שעליו הוא מוצמד. Force Magnitude מגדיר בכמה כוח האובייקט מושך אליו אובייקטים אחרים, או דוחה אותם. עבור ערך חיובי יש דחייה של האובייקט לגופים אחרים, ועבור ערך שלילי הוא מוך אותם אליו.
כברירת מחדל האובייקט מוגדר ב-Force Mode כ-constant. השדה force mode מגדיר את הדרך בה תוצג ההתנגשות. Constant למשל מגדיר שהמשיכה, או הדחייה בין האובייקטים תעשה כאשר האובייקטים יכנסו למתחם הקוליידר אחד של השני. לפעמים נרצה שההתנגשות תראה יותר ראליסטית, כלומר ככל שהאובייקטים קרובים אחד לשני יותר ככה הם ינוע אחד לשני במהירות גבוהה יותר, או במילים קצת יותר פורמליות נרצה שהגופים יקיימו את חוק הכבידה האוניברסלי: $F = G * \frac{m_1 * m_2}{r^2}$ כאשר F הוא הכוח המופעל, והוא שווה למכפלת המסות של הגופים ביחד עם G- כוח הכבידה האוניברסלי ,חלקי המרחק בין הגופים בריבוע.
למזלנו החברה של unity חשבו גם האסטרו-פיזיקאים חובבי המשחקים, ונתנו את האפשרות להגדיר את ההתנגשות בהתאם לנוסחה לעיל, כל מה שצריך לעשות הוא לשנות ה-force mode ל- Inverse squared.
אז איך משתמשים באפקטור? נוסיף לאובייקט rigidbody2D ושני קוליידרים- באחד נגדיר את גובולות האובייקט, ובשני נגדיר את מסגרת הכוח של המשיכה\דחיה, כלומר השדה שאם אובייקט אחר נכנס אליו הוא נמשך לאובייקט עם ה-pointEffector. בקוליידר של שדה הכוח נגדיר Used by effector ו- is Trigger .



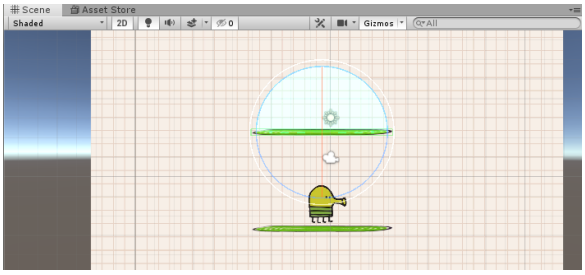
4. *Buoyancy Effector* - מגדיר את רמת "הציפה" (כמו לצוף על המים) של אובייקט כאשר הוא נכנס לאובייקט שמוגדר עם האפקטור הנ"ל.
לרוב משתמשים באפקטור הזה כדי לדמות חומר נוזלי. או ג'לי.

בעזרת ה-density מוכל להגדיר את רמת הצפיפות של הנוזל, ככל שהוא יותר נמוך האובייקט ישקע מהר יותר.
עבור צפיפות גבוה האובייקט יצוף.
ה- surface level מגדיר את פני שטח המים- ככל שהוא נמוך יותר יהיה יותר בתוך.
כמובן שהמסה של האובייקט משפיעה- ככל שהאובייקט יותר "קל" הוא כנראה יצוף ויהי פחות בתוך המים מאשר אובייקט ששוקל יותר. ניקח לדוגמא כדור חוף לעומת כדורסל, כדור החוף יצוף יותר מהכדורסל וזה יתבטא בין היתר שפחות ממנו יהיה בתוך המים, לעומת הכדורסל שחלק יותר גדול ממנו יהיה שקוע במים.
בנוסף יש את האפשרות לשלוט בזרימה של הנוזל. ובדיוק כמו באפקטורים הקודמים, flow magnitude יכול לשלוט במהירות וכיוון הזרימה.



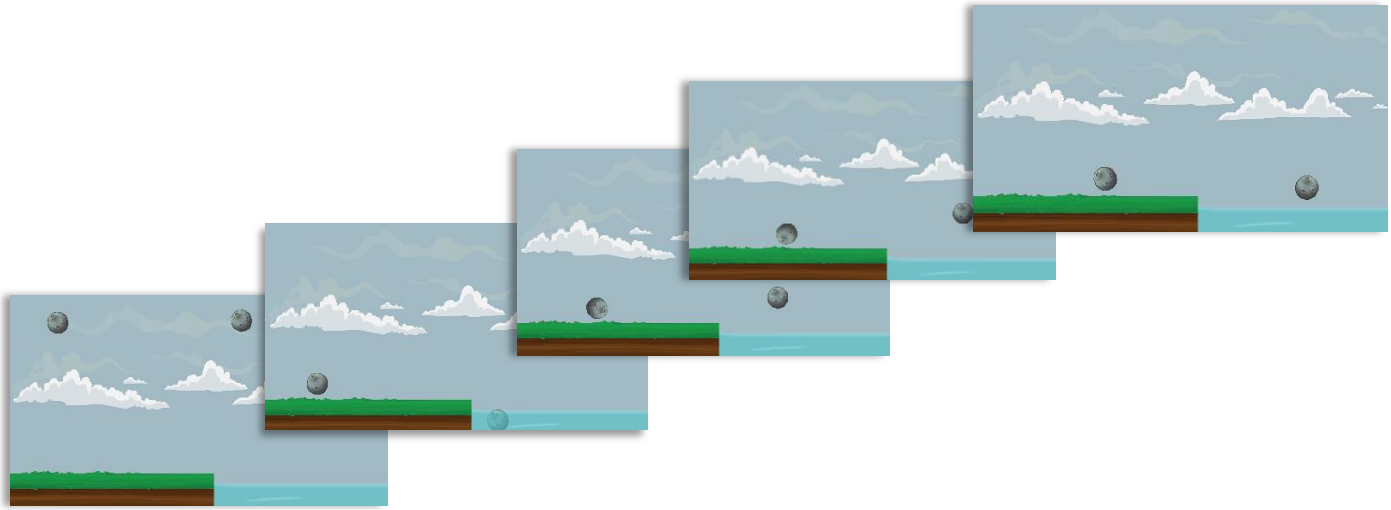
גם כאן ניצור את האובייקט בצורה דומה לקודמים. נוסיף לו קוליידר נסמן את Used by effector ו- is Trigger ונוסיף Buoyancy effector 2D.

5. *Platform Effector-* מי שמכיר את המשחק Doodle jump או את המשחק icy tower הרעיון דומה:
מצמידים אפקטור זה לפלטפורמות או משטחים למיניהם כדי שהדמות תוכל להתנגש בקוליידר של הפלטפורמה מכיוון מסוים ולא מכל מקום. למשל ב-icy tower השחקן מגיע מלמטה אבל לא נתקע מלמטה הפלטפורמה עליה הוא קופץ, אלא רק כשהוא כבר מעל לפלטפורמה הוא יכול "להתנגש" בה.
האפקטור מגיע בצורה של קשת בקוליידר אליו הוא נצמד וניתן לשנות את הזווית שלו ב-surface arc או את הכיוון שממנו תורגש ההתנגשות ב- Rotational offset.



joints2D(מפרקים)- עוד רכיב שעדיין לא יצא לנו לדבר עליו הוא מפרקים.
מפרקים מאפשר לנו ליצור קשרים שונים בין אובייקטים, למשל תנועה מקבילה, קשרי תלות כמו לגשרים ועוד.
סוגי במפרקים:
1. *Distance joints-* למפרק הראשון שנראה יש מטרה פשוטה- לשמור שני גופים במרחק קבוע אחד מהשני.

לצורך ההדגמה לקחנו שני אסטרואידים מהדוגמאות הקודמות והנחתנו כל אחד על משטח אחר- הראשון על משטח נוזלי עם Buoyancy Effector והשני עם קרקע רגילה רק שהוספנו לה Surface Effector כך שהאסטרואיד שיתנגש בה יתחיל לנוע מרגע הנחיתה .
היות ומספיק רק אסטרואיד אחד עם האפקטור ובלבד לשניהם יהיה rigidbody ,לכן רק לאחד האסטרואידים הוספנו את Distance joint וגררנו את האסטרואיד השני ל Connected rigidbody של האפקטור ,והרצנו את הסצנה:



אם לא היינו גוררים את האובייקט השני ל Connected rigidbody האסטרואיד עם האפקטור היה נשאר תלוי באוויר.
ה-Anchor (העיגול הלא מלא) מסמל את נקודת ההתחלה של החיבור בין האובייקטים. וה-connected anchor (העיגול הכחול המלא)את נקודת הסיום של החיבור.
Distance- מגדיר לנו מה המרחק הנוכחי בין האובייקטים.
Max distance only- מאפשר שהמרחק בניהם לא יעלה על המרחק המצוין ב-Distance, אבל כן מאפשר לאובייקטים להתקרב אחד לשני.

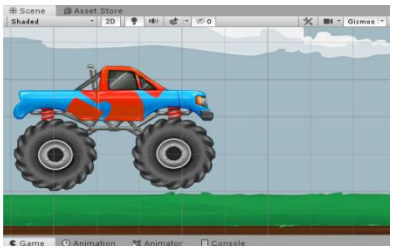
2. *Spring Joint*- בדומה מאוד ל- Distance joint גם spring joint מחזיק שדה שמורה לאובייקט באיזה מרחק להיות מהאובייקט השני. עם זאת, בעוד distance joint נשמר המרחק בצורה מוחלט, אובייקט שמחובר לאובייקט אחר ב spring joint יקפוץ הלוך ושוב לאורך אותו מרחק עד שיעצור במרחק המוגדר לו. וכמו שהשם של האובייקט מרמז הוא בא להציג אובייקט שקשור בקפיץ לאובייקט אחר.
לרוב משתמשים ב-joint במשחקים בסגנון angry birds, שרוצים שהאובייקט יקפוץ כתוצאה של שיגור מרוגטקה.
Damping Ratio –מתאר את רמת הדיכוי לתנודות הקפיץ. במילים אחרות זה מתאר באיזו מהירות האובייקט המחובר יפסיק לנוע. 0- ייקח לו הרבה זמן, 1- בקושי ינוע, אפקט זהה לשל Distance joint.
frequency- מתאר את התדירות שבו הקפיץ מתנדנד במעגלים לשנייה.

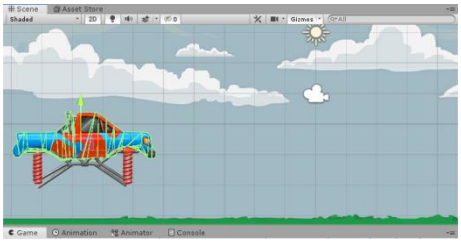
3. *Hinge Joint*- ה-Hinge קצת שונה מהשניים הקודמים. המפרק הספציפי הזה מאפשר לאובייקט עם rigidbody להסתובב סביב נקודה קבועה מראש . המפרק מחשב את הסיבוב הנכון של האובייקט כאשר כוח משפיע על הגוף הקשיח של האובייקט(הגוף קשיח צריך להיות Dynamic).
ל- hinge joint יש מספר קונפיגורציות אופציונליות שמאפשרות לנו ליצור אלמנטים דוגמת: גלגלי מים של תחנות

כוח, דלתות מסתובבות על ציר, שרשראות וכדו'.
בדיוק כמו במפרקים הקודמים: Connected Rigidbody- לאיזה אובייקט האובייקט עם האפקטור מחובר, אם הוא לא מחובר לשום אובייקט הוא נשאר להיות תלוי באוויר. Anchor ו- Connected Anchor אותו דבר מגדירים נקודת התחלה וסיום לחיבור.
Use motor- מפעיל כוח על האובייקט לנוע בכיוון מהירות (motor speed) מהירות שלילית התנועה בכיוון השלילי של הצירים, מהירות חיובית הפוך. ה-motor force מגדיר כמה כוח המנוע יפעיל כדי לגרום למומנט התזוזה במהירות שצוינה לו.
Use limit- מאפשר הגבלת הסיבוב מ360 מעלות לפחות.

4. *Slider Joint*- מגביל תנועה של אובייקט לאורך קו מסוים במרחב.
בדיוק כמו במפרק הקודם Use motor מפעיל כוח על האובייקט לנוע בקו לכיוון הרצוי תלוי במהירות, motor force מגדיר את הכוח שמופעל על האובייקט במקרה כוח מתנגד.
Use limit כאן, בשונה מהקודם, מגדיר את ציר התנועה של ה-Slider.

5. *Wheel Joint*- המפרק האחרון שלנו למדריך זה, מדמה תנועה סיבובית של גלגל שניתן לחבר לאובייקט אחר.
המפרק קיים בעיקר כדי לדמות תנועה של גלגל רכב, אולם בדומה ל-hinge joint, ניתן להשתמש בו כדי לגרום לאובייקט להסתובב במקום.
הדגמה להרצה: תחילה בנינו סצנה חדשה עם רקע וקרקע (שיש לה קוליידר), לקחנו ספריט של שילדה של רכב וספריט של גלגל. הוספנו לשילדה של הרכב rigidbody וקוליידר פוליגון. התאמנו את הקוליידר לאובייקט כך שחלק התחתון של הרכב ישמש לגלגלים:





גררנו את הספרייט של הגלגל לסצנה והוספנו לו קוליידר ו rigidbody . שכפלנו את הגלגל כדי שיהיה לנו זוג גלגלים- קדמי ואחורי. בחרנו בשני הגלגלים במקבלים בחלון הסצנה והוספנו wheel joint 2D (בגלל שסמנו את שניהם הוספה אחת מוסיפה לשני האובייקטים). בעודנו מסמנים את שני הגלגלים גררנו את השלדה של הרכב ל Connected Rigidbody של המפרק של שניהם וערכנו את ה- anchor כך שלאחר שנריץ את הסצנה השילדה תשב בדיוק על הגלגלים(בדוגמה שלנו anchors היו אחד בתוך השני, כלומר העיגול הכחול החלול בעיגול הכחול הסגור, אך לא תמיד זה כך, זה תלוי במבנה של המכונית), והרצנו לוודא שהכל עובד.
בעזרת הuse motor ניתן להגדיר כמו במפרקים הקודמים, את עוצמת התנועה וכמה כוח מופעל במקרה של התנגדות מצד כוח זר. ה-suspension שולט ברמת הקשיחות או הקפיציות של הגלגל.