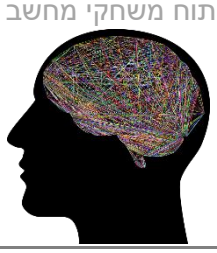


בינה מלאכותית במשחקים

חבילת המקור של הפרוייקט:
https://drive.google.com/open?id=1yoHSPrGDDUdmEEKnUgTP5VNoVdBDhdso

מבוא לבינה מלאכותית

כאשר אנו רוצים ליצור לדמות שהיא לא השחקן (NPC) התנהגות ואינטראקציה עם הסביבה, אנו נשתמש בבינה מלאכותית (Artificial Intelligence – AI) כדי לנסות ולדמות לשחקן התנהגות אנושית.

בשימוש בAI, חווית המשתמש היא הרבה יותר איכותית. הAI מעניק תחושה למשתמש שהוא משחק מול המחשב ומנסה לנצח אותו. זה גורם מרכזי לרצון של המשתמשים לחזור למשחק, ולנסות להתגבר על הקשיים שהמחשב מציב למשתמש. לכן כאשר AI במשחק מסויים נבנה בצורה לא טובה, זה יכול להיות מפתח מרכזי לכשלון של המשחק להוות חוויה שהמשתמש מעוניין בה.

AI בעצם נבנה על הפונקציונאליות שמפתח המשחקים בונה. אם נבנה AI שהוא חכם מדי, אולי לא יהיה אפשרי לנצח את המחשב. לרוב זה דבר לא טוב. אם נבנה AI טיפש מדי, המשתמש ישתעמם מהר מן החוויה. גם לא טוב. לכן תחום הAI הוא תחום מרכזי במשחקים המבוססים על חווית משתמש יחיד או רבים מול מחשב, ותכנון הAI צריך להיות מדוייק. אחרת, חווית המשתמש תהרס.

מנוע הAI של UNITY

ספריה בשם AI מרכזת את כל הנצרך ליצירת AI במשחק שניצור בסביבת UNITY.

NavMeshAgent

כל האובייקטים והפונקציות במנוע של UNITY AI, קשורות לרכיב בשם NavMeshAgent – הרכיב האחראי על ההגדרות המקשרות בין NPC לבין הפעילות שלו בסביבת הNavMesh שלמדנו עליה בשיעור הקודם.

הפונקציות שהספרייה מספקת לנו, הן פונקציות חישוביות שקשורות למסלול והסביבה של הAI. הפונקציה המרכזית שבה אנו מחשבים מסלול היא הפונקציה Set Destination:

קביעת המטרה

כדי להודיע לשחקן לאן להגיע, אנחנו נשתמש בפונקצייה .NavMeshAgent.SetDestination. המנוע של יוניטי יחשב את הכיוון ואת המרחק שהשחקן ינוע.
הוא יעשה זאת תוך התחשבות באובייקטים השונים הנמצאים בדרך.
הוא בעצם מחשב מסלול בהתחשב בNevMesh שבסצנה, שמראה איפה נמצאים קוליידרים שיפריעו לNPC להתקדם במסלול.
לכן יש לעקוף אובייקטים אלה, הנקראים בדרך כלל "מכשולים".

אז אם נרצה לגרום לאובייקט מסויים לעקוב אחרי השחקן, מה נעשה?
נוכל למשל לייבא את האובייקט של השחקן, ולשלוח לSetDestination את הtranform.position של השחקן.

למשל כך:

```
using UnityEngine.AI;     // לא לשכוח

class Enemy {

private NavMeshAgent navMeshAgent;
private Player player;
private Vector3 dest;

private void Start() {

    nevMeshAgent = this.GetComponent<NavMeshAgent>()   ; // לא לשכוח לבדוק שהפוינטר תקין

   player = FindObjectOfType<Player>();     // לא לשכוח לבדוק שהפוינטר תקין

}

private void Update()  {

    nevMeshAgent.setDestination(_player.transfom.position);

}
```

התוצאה היא שכיוון האובייקט תמיד יהיה הכיוון שבו השחקן נמצא.

נניח ונרצה ליצור דמויות אחרות שנוכל לשלוט עליהם, ולהגיד להם לאיפה ללכת על ידי לחיצה על הכפתור הימני בעכבר.

איך נעשה זאת?

דבר ראשון נרצה לדעת איזה מיקום סימננו על המפה. פה נכנס כלי חשוב מאוד שכבר דיברנו עליו בשיעור על פיסיקה – שליחת קרניים - RayCast.

נראה את הקוד הבא:

```
If(Input.GetMouseButtonDown(1))
{
    Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
    RaycastHit hitInfo;
    physics.RayCast(ray, out hitInfo)
    navMeshAgent.SetDestination(hitInfo.point)
}
```

אם נשתמש בקוד הזה על כל NPC הוא יציית לפקודות שלנו.
זאת התחלה ליצירת Companion (שותף) כך שנוכל לשלוט בטקטיקה שלו בזמן משחק.
בהמשך נלמד קצת על ירי של NPC, וככה נוכל גם לגרום לCompanion שלנו לירות על NPC אחרים ובכך לעזור לנו לנצח במשחק!

בינה מלאכותית לאויב

במשחקי FPS (First Person Shooter) רוב הNPC נחשבים כאויבים לשחקן. הם בדרך כלל ינסו לירות על השחקן ולגרום לו להפסיד במשחק.
מה אנחנו צריכים בעצם שהאויב יעשה? הדברים הבסיסיים שאויב עושה זה: רץ, מתחבא, יורה/תוקף. כמובן שזה תלוי במשחק ובצורך.
ישנם דברים כמו התנהגות בקבוצה, סולם העדפות לאויב, (האם לירות על השחקן? או להשמיד מטרה אחרת?), להיות תוקפני? או הפוך, דווקא להיות פחדן...

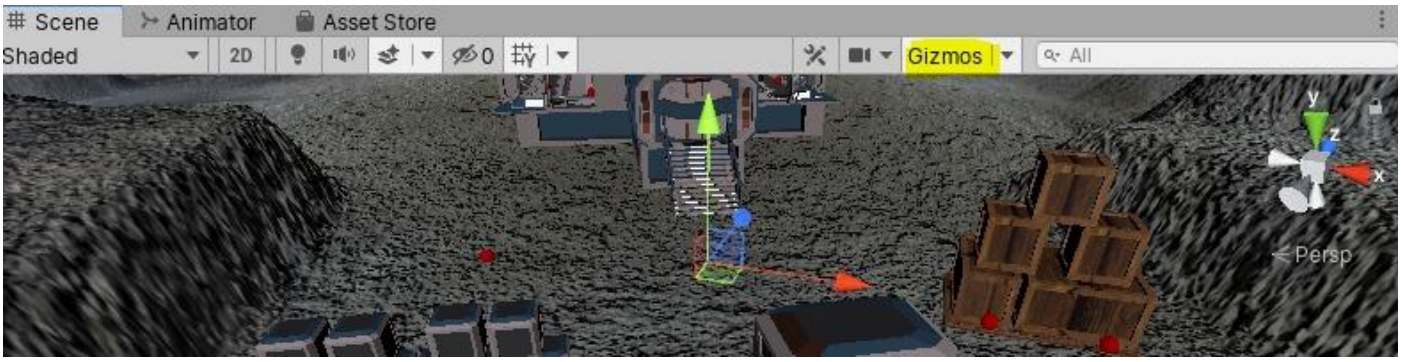
בסופו של דבר כל ההתנהגויות שניצור לNPC שלנו הן סינטטיות ונגרמות על ידי הלוגיקה שבנינו בקוד שלנו.
נראה מספר דוגמאות, שיעידו כי כל התנהגות נופל על לוגיקה ופרמטרים:

כאשר אנו רוצים להגדיר לאויב לאיפה ללכת, אנו עושים זאת על ידי setDestination. אבל מה המקום אליו נגיד לו ללכת?
אל השחקן? אולי, זה תלוי בכם.
אבל מה אם נרצה שהאויב שלנו יוכל לבחור בעצמו לאן ללכת?
במשחק שנבנה כאן, נשתמש בterrain שיצרנו במדריכים הקודמים, ונמקם בו חיילים.
החיילים האלה ירוצו על גבי המפה, ויבחרו לאיזה מחבוא/או נקודה נרוץ הלאה.
לכל נקודה כזו נוכל לקרוא בשם Target והוא יהיה בעצם אובייקט ריק.

ניצור אובייקט ריק בשם target וניצור לו script. נשתמש בGizmos כדי שנוכל לראות את הtarget על הscene view.

מחלקת Gizmos:

נועדה לעזור למשתמש על ידי מתן מראה ויזואלי לרכיבים ואובייקטים שונים בסצנה שלנו. כדי להפעיל את הGizmos נצטרך ללחוץ על כפתור הGizmos בחלון הscene view.



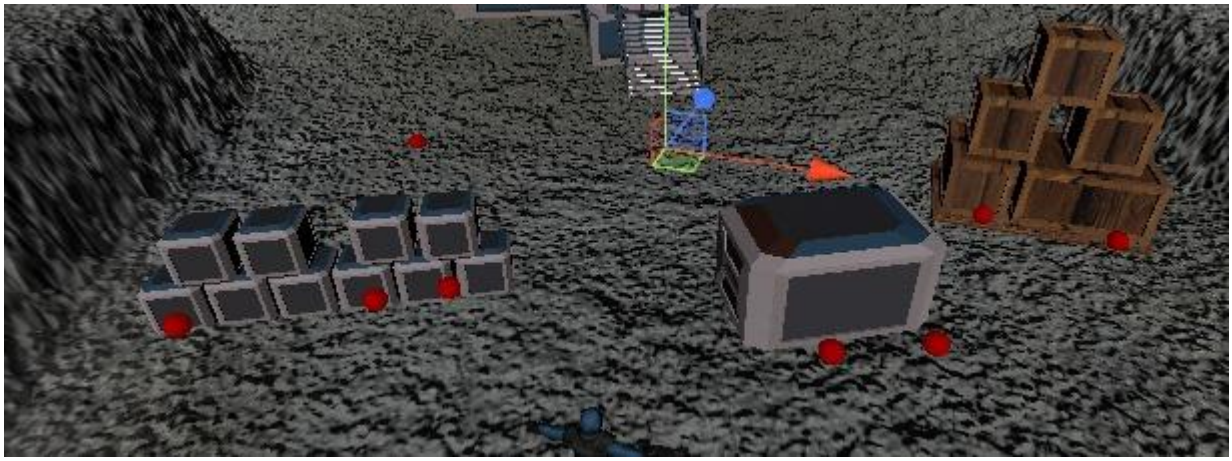
בוא נראה את הקוד הבא לאובייקטי הTarget שלנו:

```
public class Target : MonoBehaviour {

    private void OnDrawGizmos() {

        Gizmos.color = Color.red;

        Gizmos.DrawSphere(transform.position, 0.3f);

    }
```

כל מה שעשינו בקוד הזה, זה לצבוע כל אובייקט Target בצבע אדום.

נפזר כמה אובייקטים כאלה בנקודות מפתח שנרצה על גבי אזור המשחק שלנו, ונשרשר את כולם לאובייקט ריק Targets, אך ורק בשביל הסדר בלשונית ההיררכיה.

זה צריך להראות כך:



כעת כשאנחנו יודעים לאן האויב שלנו אמור ללכת, בוא ניצור אותו!

הורידו את החבילה בשם ToonSoldier_demo מהחנות של unity.
בתוך התיקייה של הAsset הנ"ל, נכנס לתיקיית models ושם נגרור את האובייקט ToonSoldier_demo לאובייקט ניתן את השם Enemy.
אם נסתכל עליו עכשיו בscene view נראה משהו ממש מצחיק.
לכן נכנס עכשיו לתת נושא חדש ונסביר את הדברים.

אנימציה לדמויות

בשלב זה נלמד איך להתמודד עם אנימציה בUnity, ולא נלמד איך ליצור אנימציה בעצמנו.

ישנם תוכנות ליצירת אנימציות כגון Blender, אך על זה אפשר ליצור קורס שלם בפני עצמו.
במקום זה נראה מה עושים עם אנימציה מוכנה בunity.

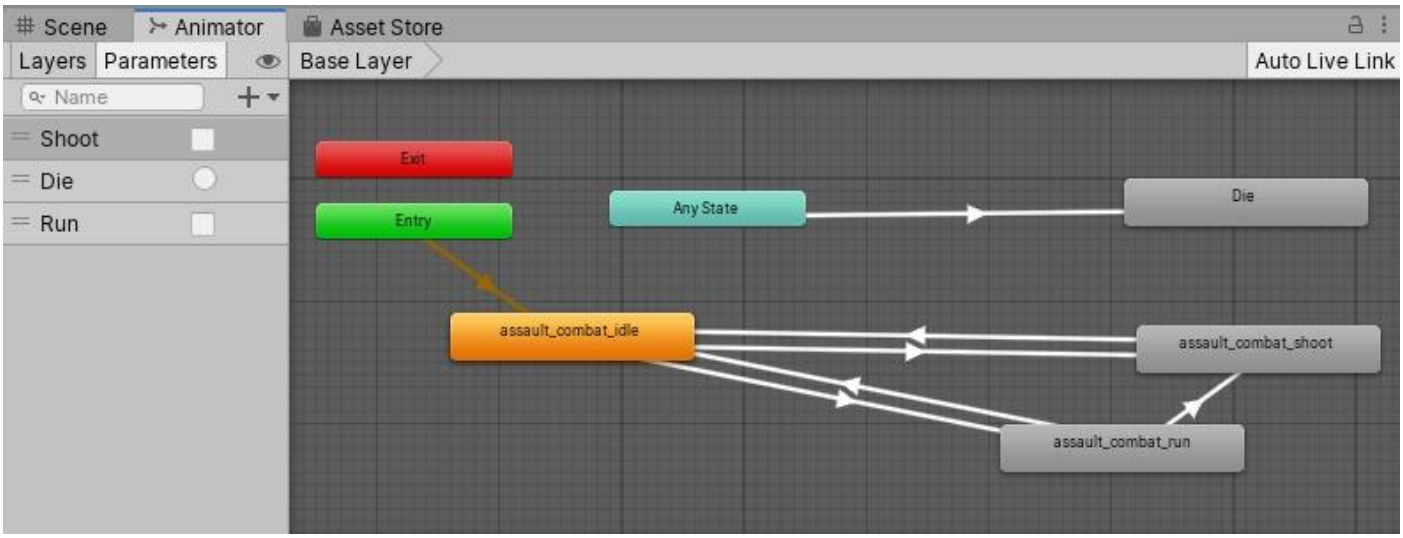
דבר ראשון, נראה שהשחקן שלנו עומד בצורה מוזרה.

כן... אז יש שם לעמידה הזו. זה נקרא T-Pose.

T-Pose זוהי צורת הברירת מחדל שבה עומד אובייקט של שחקן לפני שנפעיל את הAnimator/לפני שהNPC מתעורר.

זו היא נקודת התחלה שממנה מתחילים לבנות את התנועות השונות של הדמות.
כאשר הAnimator מתחיל לפעול, נראה שהדמות תזוז לאנימצית ברירת המחדל של הAnimator.

כעת נסתכל על האנימציות השונות במכונת המצבים של הAnimator.
נשים לב כי עם השחקן שהורדנו, קיימת מכונת מצבים כפי שמופיע בתמונה מטה.



כדי להוסיף מצב חדש, פשוט נגרור מתיקיית האנימציות של הasset שלנו את האנימציה לחלון מכונת המצבים.
כעת נבחר באנימציה של idle כמצב default ע"י לחיצה על המצב עם כפתור ימני בעכבר ובחירה בset as layer default State.

לשם רענון, נזכר קצת מה הם החיצים השונים שנמצאים במכונת המצבים, ולמה יש מצבים בצבעים שונים.

המצבים האפורים – מצבים רגילים.
המצב הצהוב – Default State.
המצב הירוק והאדום – מצב כניסה ויציאה שקיים בכל animator, מצב הכניסה מוביל למצב הדיפולטיבי, ובדרך כלל מצב היציאה מגיע כאשר האובייקט כבר לא נמצא בסצנה.
(למשל דמות שרואים הולכת לאורך מסלול מסויים, ברגע שהדמות תצא מטווח הראייה האפשרי של השחקן, היא תכנס למצב Exit ומשם לT-Pose).
המצב הכחול – AnyState, מאפשר ליצור מעבר מכל מצב שאנחנו נמצאים בו עכשיו אל מצב חדש (למשל, רוצים שהאויב יעבור לאנימציה של "מוות", במקום להוסיף חיצים מכל המצבים למצב "מוות", מוסיפים חץ אחד מ AnyState).

החיצים- כל חץ בעצם נקרא Transition (מעבר). זה בעצם אומר, שאם קורה משהו מסויים, נעבור ממצב אחד למצב אחר.

למשל החץ שעובר מהמצב idle (הdefault state) למצב Run. כאשר הbool בשם run יהיה במצב true, נעבור בחץ (טרנזאקציה) הזה למצב run ממצב idle. וכעת תתחיל אנימציה של ריצה.
כאשר נסמן טרנזאקציה כזו עם העכבר, הinspector יראה כך:

נראה כי נוכל ליצור condition.
condition זה התנאי שאם הוא מתקיים יתבצע המעבר (transision).
במקרה שלנו, הגדרנו כי כאשר run יהיה שווה true, יתבצע המעבר.

איך בסקריפט שלנו ניגש למשתנים האלה?
ניצור אובייקט מסוג Animator ונאתחל אותו על ידי GetComponent.
לאחר מכן ניגש אל האובייקט ונפעיל את הפונקציה SetBool.

```
animator.SetBool("Run", true);
```

דברים נוספים:
1. בלשונית parameter נוכל להוסיף פרמטרים שונים, כגון bool וtriger על ידי לחיצה על כפתור ה+/
2. נוכל ליצור מעבר חדש על ידי לחיצה על מצב עם כפתור ימני בעכבר, ושם לבחור באופצייה Make Transition.

כעת נסו ליצור את מכונת המצבים המופיעה מעלה, ותקבלו את האנימציה הרצויה לאויב שלנו.

לוגיקה לדמויות

כעת נחזור לנושא הפונקציונאליות ב-AI.

ראשית ניצור דרך לגרום לאויב לבחור מטרה כלשהי על הסביבה שלנו.

ניצור אובייקט target ומערך של target לתוכו נכניס את כל אובייקטי הtarget שבפרוייקט.
איך נעשה זאת?

בדרך הבאה:

```
using UnityEngine.AI;


public class Enemy : MonoBehaviour {

    private NavMeshAgent navMeshAgent;

    private Target[] targets;

    private Target target;

      private void Start() {

                Targets = FindObjectsOfType<Target>();

      }

      private void Update()      {

      }

}
```

כעת נרצה לגרום לשחקן לבחור טארגט כלשהו.

איך נעשה? נוסיף גם בstart וגם בupdate:

```
      target = targets[Random.Range(0, targets.Length - 1)];

      navMeshAgent.SetDestination(target.transform.position);
```

אז מה עשינו בעצם? גרמנו לאויב שלנו לבחור מספר רנדומאלי שלם, בין 0 לגודל המערך (פחות אחד), וזה יהיה האיבר במערך שיצרנו. כל איבר במערך הוא target, ואמרנו לNPC שלנו, לך לכיוון הtarget.

אם תפעילו כעת את המשחק, השחקן שלנו יתחיל לרחף לכיוון שהוא בחר.

אם נרצה נוכל להפעיל את האנימציה שלנו כדי שהשחקן יתחיל "לרוץ".
לכן נכתוב

```
animator.SetBool("Run", true);
```

וכאשר הNPC יגיע אל המקום אליו הוא רצה להגיע, נפסיק את האנימציה כך:
בתוך Update (או שנוכל ליצור פונקציה חיצונית)-

```
if (!navMeshAgent.hasPath) {

                animator.SetBool("Run", false);

 }
```

כעת זה נראה כאילו הNPC מתרוצץ לו ממקום למקום, וקשה לנו לראות כאשר הוא מגיע ליעד שהוא הציב לעצמו.

לכן נרצה ליצור איזשהו מנגנון שיגרום לו להמתין במקום לפני שיבצע את הבחירה הבאה שלו.
נוכל ליצור coroutine חדש, ופשוט לגרום לפונקציה לחכות כפי שלמדנו.

נוכל ליצור גם אובייקט float שתסמן את הזמן עד המעבר לפונקציה הבאה. וכל פעם לחסר אותה בTime.DeltaTime.
אם נגדיר 7f, לא נכנס לפונקציה 7 שניות וכו'.

```
private float nextState;//משתנה מחלקה- לא לשכוח להצהיר!

nextState -= Time.deltaTime;

if (!navMeshAgent.hasPath)

{

animator.SetBool("Run", false);

if (NextState <= 0)

                {

                        target = _Targets[Random.Range(0, _Targets.Length - 1)];

                        navMeshAgent.SetDestination(_target.transform.position);

                        nextState = Random.Range(7f, 15f);

                        animator.SetBool("Run", true);

                }

}
```

אז נבדוק קודם, האם הNPC הגיע למקום היעד? או במילים אחרות, האם נשאר לו עוד מסלול לעבור עד ההגעה ליעד?
אם לא, עצור את אנימציית הריצה.
ולאחר שהזמן ב nextState יגיע לאפס, נבחר יעד חדש לNPC שלנו.

במשחקים מסויימים, אולי נרצה לתת לNPC שלנו להסתובב בחופשיות על המפה שלנו בלי להיות כפוף לנקודות מסויימות על המפה. אפשר לדוגמא לעשות זאת בדרך הזאת:

```
private Vector3 GetRandomDir(){

   float x = UnityEngine.Random.Range(-1f, 1f);
```

```
    float z = UnityEngine.Random.Range(-1f, 1f);

    Vector3 dir=new Vector3(x,0,z).normalized;



    return startinPosition + dir * Random.Range(10f, 70f);

}
```

פשוט נחזיר את האובייקט Vector3 ונפעיל את פונקציית SetDestination.

כעת אנו רואים שיש לנו NPC שבכוחות עצמו מסתובב לנו על המפה ומחליט לאן ללכת.

הבעיה שכעת נרצה גם שהאויב יעשה איתנו אינטראקציה כלשהי. לעת עתה הוא ממש אנטיפת!

דבר ראשון נרצה שהוא בכלל יביט לכיוון שלנו כשנתקרב אליו.

איך נעשה זאת?

דבר ראשון ניצור שלושה משתני מחלקה:
הראשון הוא gameObject של Player שיהיה serializeField, ונכניס לתוכו את הplayer שלנו.
השני הוא Vector3 שייצג את המיקום של השחקן שלנו בכל שלב על המפה.
השלישי הוא float שייצג את המרחק שהNPC "רואה".

```
private float lookRadius = 10f;//משתנה מחלקה

private Vector3 _playerPos;//משתנה מחלקה

[SerializeField]

private GameObject _player;//משתנה מחלקה

float distance = Vector3.Distance(_playerPos, transform.position);

private void Update()

    {

        _playerPos = _player.transform.position;

        float distance = Vector3.Distance(_playerPos, transform.position);

        if (distance <= lookRadius)

        {

            FacePlayer();

        }

}
```

```
private void FacePlayer()
    {
        Vector3 direction = (_playerPos - transform.position).normalized;
        Quaternion lookRotation = Quaternion.LookRotation(new Vector3(direction.x, 0 , direction.z));
        // transform.rotation = lookRotation;
        transform.rotation = Quaternion.Slerp(transform.rotation, lookRotation, Time.deltaTime * 5);
    }
```

הסבר:

בכל שלב אנו בודקים את המיקום של השחקן על גבי המפה.
אם הוא נמצא בתוך טווח הראייה של הNPC (בדקנו זאת ע"י שימוש בפונקציה של מחלקת Vector3), נעבור לפונקציית FacePlayer.

בפונקצייה FacePlayer אנו מוצאים את הכיוון הכללי של השחקן על המפה ע"י שימוש בפונקציית הנירמול.

לאחר מכן אנחנו משתמשים באובייקט הQuaternion על מנת להגדיר סיבוב.
כדי לעשות זאת נשתמש בפונקציית LookRotation וניתן לה את הכיוון של הצירים השונים עם אובייקט Vector3.

לבסוף נצטרך להגדיר את הtranform.rotation. נוכל פשוט לתת לו את אובייקט lookRotation, אבל אם נרצה ליצור תנועה חלקה יותר, נוכל להגדיר על ידי פונקציית Slerp את זמן גידול האפסילון של כיוון הרוטציה. כך האויב יסתובב בהדרגה.

להרחבה בנושא:

https://docs.unity3d.com/ScriptReference/Quaternion.Slerp.html

גם כעת נוכל לגרום לNPC לירות עלינו כשאנחנו נהיה בטווח שלו.
ממש כמו בplayer נוסיף לו muzzleFlash וBulletHole.

גם נוכל להוסיף לו AudioSource ולהפעיל אותו עם AudioClip שידמה ירי.

כך תראה הפונקציה:

```
private void ShootPlayer() {
        animator.SetBool("Shoot", true);
        audioSource.Play();
        muzzleFlash.SetActive(true);
```

```
//position ray casted from

Ray rayOrigin = new Ray(transform.position + Vector3.up * 1.5f, transform.forward);

RaycastHit hitInfo;

if (Physics.Raycast(rayOrigin, out hitInfo))    {

        GameObject hitMarker = Instantiate(_bulletHole, hitInfo.point,
Quaternion.LookRotation(hitInfo.normal)) as GameObject;



            Destroy(hitMarker, 1f);

}
```

אז שלחנו rayCast שיוצא בערך מהמיקום של הרובה של האויב.

מה אם נרצה למשל לדעת שפגענו בשחקן? אולי כדי להוריד לו מהחיים? איך נעשה את זה?

פשוט נכתוב את השורה הבאה:

```
If(hitInfo.collider.GameObject.tag == “Player)
//תעשה משהו
```

גם באותה דרך נוכל לגרום לcompanion שלנו לבדוק האם הוא פגע בשחקן שלנו.
נניח ולא נרצה לתת לcompanion לפגוע בשחקן.
קודם נבדוק האם הRayCast שלנו פגע בשחקן. נניח והוא פגע, נוכל לוותר על כל האנימציה של הירי, וזה יראה כאילו הcompanion שלנו מעולם לא ירה על השחקן.

במטלה תצטרכו למצוא דרכים יצירתיות לפתח את הAI שלנו.
כעת הוא די טיפש, אבל אתם תצטרכו לגרום לו להחכים מעט.

מדריכים מומלצים נוספים

Navigation Mashes in unity (by Table Flip Games):
https://www.youtube.com/playlist?list=PL8lV_joQZ5sfqiNwoJcokJlcrgplW8uSs

Unity Shooting Scene Tutorial (by DitzelGames):

https://www.youtube.com/playlist?list=PLA6Gf0nq2Gh47crwfsfiSZ5_fVYnkfiU8

Final State Machines in Unity (by Table Flip Games):

https://www.youtube.com/watch?v=21yDDUKCQOI&list=PL8lV_joQZ5sczN_xHOEXEmfSIt3gYr1Rh

Unity3D AI with State Machine (FSM), Drones, and Lasers:
https://www.youtube.com/watch?v=YdERlPfwUb0

How to Make an RPG in Unity (by Brackeys):
https://www.youtube.com/playlist?list=PLPV2KyIb3jR4KLGCCAciWQ5qHudKtYeP7

RPG Core Combat Creator: Learn Intermediate Unit C# Coding (מומלץ):
https://www.udemy.com/course/unityrpg/learn/lecture/14204210

Making a Multiplayer FPS in Unity(by Brackeys):
https://www.youtube.com/watch?v=UK57qdq_lak&list=PLPV2KyIb3jR5PhGqsO7G4PsbEC_Al-kPZ

סיכם: מיכאל למברגר