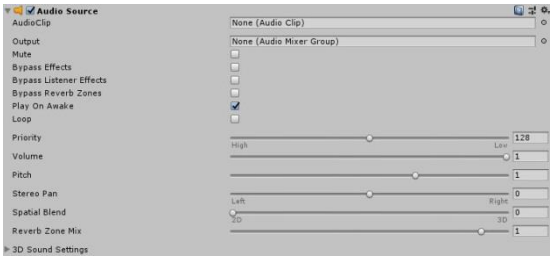
קבצי שמע

אפקטי קול הם חלק אינטגרלי בכל משחק שמכבד אותו (למעט משחקים שהקהל יעד שלהם הוא חרשים).
החשיבות באפקטים קוליים ושימוש במנגינות רקע שהם יכולים לשלוט ברגשות השחקנים ולהכתיב את הטון של הסיפור שמלווה את המשחק. יש משחקים שהסאונד-טראק שלהם נהיה כל כך מהותי במשחק שאנשים זוכרים בעיקר אותו.
קחו לדוגמא משחק כמו סופר-מריו, אם נעשה סקר שוק מה אנשים זוכרים יותר- את הסיפור רקע של המשחק או את הסאונד-טראק שלו, סביר להניח שיותר אנשים יזכרו את המוזיקה ולא את הסיפור על אף שהוא מהות המשחק (לכאורה).
השימוש בסאונד ב-unity מופעל ע"י שני דברים: **AudioSource-** שאחראי לנגן את הקבצים בזמן המשחק. במשחקי תלת-ממד הצליל יכול להתכוון לפי המרחק, נניח דמות רחוקה ממך אתה תשמע אותה פחות מאשר אם תהיה לידה;
ו- **AudioClip** – הקובץ שמע (למשל mp3) שאותו מנגנים.



ניתן לראות מהתמונה לעיל כי ל- AudioSource יש את האפשרויות הבאות:
output- להיכן יוצא הקובץ. כברירת מחדל הקובץ שלנו יוצא לאיזשהו Audio Listener שהוא כמין מכשיר מיקרופון. לרוב הוא מחובר למצלמה. אפשרות נוספת היא להוציא את הסאונד ב-Audio Mixer בשביל להפיק יותר אפקטים קוליים
Mute- משתנה בוליאני שמגדיר האם הסאונד נשמע כעת או לא. הוא לא עוצר את נגינת הסאונד אלה רק "מנמיך" את עוצמת הקול ל-0.
Bypass effect- דרך מהירה להפעיל\לכבות את כל האפקטי קול של ה-AudioSource.
Bypass listener effect- אותו דבר רק על ה-Audio Listener .
Play on Awake-משתנה בוליאני שמאפשר להפעיל את הקובץ ישירות כאשר הסצנה מוטענת למסך. אם לא נסמן את זה נצטרך להפעיל את האודיו-קליפ בקוד דרך המתודה ()Play.
Loop- מנגן את הצליל בלופים עד שיבטלו את הצליל או את המשתנה loop .
Priority- קובע את חשיבות הצליל יחסית לצלילים אחרים בסצנה. 0- החשיבות הכי גבוהה ו- 256- החשיבות הכי נמוכה(הברירת מחדל היא 128). משתמשים ב-0 לרצועות מוזיקה בדר"כ.
Volume- עוצמת הסאונד. מוגדרת להיות כמה המוזיקה "רחוקה" מאתנו (מה-Audio Listener) כל יחידה שווה בערך מטר.
Pitch- מהירות הסאונד .1-מהירות נורמלית, מתחת לזה המהירות איטית יותר ומעל זה המהירות גבוה יותר, המהירויות מוצגות ככפולות של המהירות המקורית, כך למשל מהירות 1 היא כאילו פי 1 מהמהירות המקורית שזאת אותה מהירות.
Stereo pan- קובע להיכן הצליל קרוב יותר- לאזור הימני או השמאלי.
Spatial Blend- קובע כמה השפעה יש למנוע התלת-ממדי על ה-AudioSource.

מעצם היות ה-AudioSource רכיב של unity, כלומר יורש מ- monobehavior,הוא יכול להיצמד ישירות לכל GameObject של unity.
חלק מרכזי מהפעלת אפקטים קוליים הוא השימוש באיזשהו טריגר שיפעיל אותם, איזשהו תנאי שיתקיים שלפיו יופעל הקובץ (למשל פגיעה ע"י לייזר, תחילת משחק וכו').

מוזיקת רקע-

נתחיל בדבר הפשוט יותר לעשות- מנגינת רקע.
בשביל מוזיקה שמתנגנת ברקע נצטרך איזשהו אובייקט שיכיל בתוכו את האודיו-קליפ. ניצור אובייקט ריק חדש ונוסיף לו את הרכיב AudioSource. למי שלא זוכר, כדי להוסיף רכיב חדש צריך ללכת ל- Add Component באינספקטור, ולחפש בשורת חיפוש את שם האובייקט אותו אנחנו רוצים.
לתוך הרכיב שהוספנו נגרור את קובץ שמע לשורה Audio Clip, ונוודא ש-play on awake מסומן (שברגע שמופעל האובייקט משחק מופעל גם האפקט קול), ושגם loop מסומן, כי אחרי הכל אנחנו רוצים שהשיר יתנגן במשך כל הסצנה.
וברמת העיקרון זהו, לא צריך אפילו להתעסק בקוד. נשמור את הסצנה ונריץ את המשחק לוודא שהכל עובד.
ולפני שנמשיך ניתן למצוא ולהוריד מוזיקת רקע נהדרת למשחק דרך האתר YouTube. למי שלא מכיר, יש מלא אתרים שמאפשרים להוריד את התוכן של הסרטון או לפחות את הפס-קול שלו בפורמט mp3. דוגמא לאתר:

/https://2conv.com/en4/youtube-mp3

אפקטים קוליים-

יש כמה גישות באשר איך לעשות את האפקטים הקוליים. לפי הגישה הראשונה שצורכת מינימום של קוד היא להוסיף לכל אנימציה ייעודית (או אובייקט ייעודי) את הצליל שמתאים להם שיופעל ברגע שהם נוצרים. למשל נצמיד לאנימציה של הפיצוץ סאונד של פיצוץ כאשר הוא נוצר, ואפילו לא נצטרך להיכנס לקוד היות ויש לנו את המשתנה Play on awake באינספקטור, כך שהצליל יופעל עם רגע היווצרות האובייקט. במקרים בהם יש לנו כמה אפקטים קוליים לאובייקט מסוים, לדוגמא דמות שפעם בעשר שניות מזכירה שצריך למהר, ופעם בחצי דקה מזכירה לו מה המשימה. נוכל להשתמש במערך של אודיו-קליפים וכל פעם יוטען לAudioSource קליפ אחר. החיסרון כאן ברור- אין אפשרות לערוך כל קליפ בנפרד, יוצא שלכל הקליפים יהיו את אותם מאפיינים, מה שלא יעזור לנו אם נרצה שלכל קליפ יהיה את הערכים שלו. אפשרות אחרת היא לעשות מערך של AudioSource, ואז נטעין למאורע את האחד הרלוונטי לו מהמערך.

גישה שניה, צורכת קצת יותר מאמץ אך שווה את זה לטווח הארוך, היא ליצור איזשהו Audio Manager שינהל לנו את הסאונד המתאים לכל מאורע. כך באירוע מסוים נבקש ממנהל הסאונד שיפעיל לנו צליל שמתאים לאותו אירוע. זה יעזור לנו לשמור על קוד יותר גנרי-יהיה לנו מקום מסודר לכל קבצי הסאונד שלנו ולא נצטרך לקפוץ בין אינספקטורים של כמה אובייקטים כדי לשנות צליל מסוים. כמו כן זה מאפשר לשמור על אותם צלילים בכמה סצנות. כפי שנראה בהמשך במעבר בין סצנה לסצנה (או במעבר בין שלבים) לא נשמרים לנו אותם נתונים אלא אם הגדרנו מראש את האובייקט כ"לא נמחק בין סצנות" (כמין אובייקט סטטי).
אם לא נשתמש במנהל, יהיה לנו קשה יותר לשמר כמה צלילים שיתנגנו לאורך כמה סצנות. נצטרך לעבור אובייקט אובייקט ולהגדיר אותו במיוחד. ובמנהל סאונד פשוט נגדיר אותו ככה וזה יספיק לנו. אבל איך ניצור את ה-Audio Manager ?

Audio Manager-

תחילה ניצור אובייקט ריק חדש עם השם Audio Manager ונחבר לו סקריפט.
אנחנו רוצים לשלוט בקבצי האודיו ולכן נצטרך להשתמש במרחב שם ייעודי של unity שמטפלת בקבצי אודי:

```
Using UnityEngine.Audio;
```

הרעיון במנהל האודיו הוא שנוכל להוסיף ולהפחית צלילים בקלות לרשימה תוך כדי שימוש. ואז כאשר נצטרך להשתמש באחד הצלילים נחפש את אותו הצליל שמתאים לאירוע מתוך הרשימה שהכנו מראש.
בשביל שנוכל לשלוט במידע שנכנס לנו לתוך הרשימה בקלות ובלי סרבול של שימוש ב-AudioSource, ניצור מחלקה חדשה שתהווה מעטפת לאודיו קליפים. ניצור סקריפט חדש ונקרא לו Sound לצורך הפשטות.
צריך להכליל במחלקת Sound שלנו גם את הספרייה הייעודית של unity לקבצי אודיו כפי שעשינו ל-Audio Manager.
אין צורך שהמחלקה תירש מ-monobehavior אז נמחק את שורת הירושה, וכנ"ל המתודות ()start ו- ()update .
למחלה הייעודית שאנחנו בונים נצטרך כמה משתנים:

1. AudioClip- שאותו אנחנו מנגנים
2. משתנה float שאחראי על עוצמת הקול (volume)
3. משתנה float שאחראי על מהירות הסאונד (pitch)
4. משתנה bool שיגדיר לנו האם להשמיע את הצליל בלופים (loop)
5. מחרוזת שמייצגת את שם האובייקט עליו אנחנו עובדים (name)

בשביל הפשטות נגדיר את המשתנים כ-public שיהיה לנו קל לעבוד איתם (ניתן להגדיר כprivate ולהוסיף להם getter ו-setter). כדי להשתמש ב-unity במחלקה חיצונית (שלא מחוברת לשום אובייקט ולא יורשת מ- monobehavior) אנחנו צריכים לסנכרן את המחלקה עם המערכת. אפשר לסנכרן מחלקות ע"י ה- casting הייחודי: [System.Serializable] בראש ההצהרה של המחלקה.
בשביל להקל אף יותר, ל-unity יש את האפשרות להגדיר למשתנים כפתורים מיוחדים באינספקטור עוד בקוד ע"י casting מסוים, למשל כדי להפוך את המשתנה volume באינספקטור לכפתור הזזה עם טווח מסוים למשל בין 0 ל-1 נוכל להשתמש ב-casting: [Range(0f,1f)] כך נוסיף הגבלה למשתנה מבלי להיכנס ל-if מיותר, אותו דבר אפשר לעשות ל-pitch בערכים בין 0.1f ל-3f (כך הערכים ב-AudioSource).
בשביל להפעיל כל אודיו-קליפ אנחנו צריכים משתנה מסוג AudioSource, נגדיר אותו כ-public אך כדי לא לראות אותו באינספקטור אם צורך בכך (כי זה סתם תופס מקום מיותר), נצמיד את ה- casting: [HideInInspector] .
מבחינת סינטקס זה יראה כך:

```
using UnityEngine;
using UnityEngine.Audio;
using System.Collections;
using System.Collections.Generic;

[System.Serializable]
public class Sound
{
    public string name;
    [Range(0f,1f)]
    public float volume;
    [Range(0.1f,3f)]
    public float pitch;
    public bool loop;
    public AudioClip clip;
    [HideInInspector]
    public AudioSource source;
}
```

נחזור ל-Audio Manager: נצטרך משתנה שיאחסן בתוכו את כל משתני ה-sound שנשמור. ניתן להשתמש בכל דבר שיורש מcollection בעדיפות על dictionary (ה-Hash Table של #c) ששם פשוט נצמיד את השם של הקובץ כמפתח לאובייקט מסוג Sound, אך הבעיה בשימוש בdictionary היא שלא לא ניתן לראות את האובייקטים שנכנסים אליו דרך האינספקטור, או יותר נכון זה דורש הרבה יותר עבודה. לעומת מערך פשוט ,שאומנם פחות יעיל אך **הרבה** יותר קל לעבוד איתו .
אנחנו צרכים לעבור על כל משתני הסאונד ברשימה שלנו ולאתחל להם את ה-AudioSource כדי שנוכל לנגן את האודיו-קליפים ביתר קלות. נוכל לאתחל אותם באמצעות לולאת for each שתעבור עליהם אחד אחד ולהתאים את הערכים של ה-AudioSource של משתני סאונד שלנו, לערכים שהגדרנו להם מראש.
את האתחול נבצע דווקא לא במתודה start(), אלא במתודה Awake(), שהיא דומה מאוד ל-start() אך שהיא מתחילה עוד קודם לכן (מתחילת הסצנה ולא מיצירת האובייקט). מבחינת סינטקס זה יראה כך:

```
public class AudioManager : MonoBehaviour
{
    public Sound[] sounds;

    void Awake()
    {
        foreach(Sound s in sounds)
        {
            s.source = gameObject.AddComponent<AudioSource>();
            s.source.clip = s.clip;
            s.source.volume = s.volume;
```

```
            s.source.pitch = s.pitch;
            s.source.loop = s.loop;
        }
    }
  . . .
}
```

כמובן שבמהלך המשחק אנחנו צריכים להריץ את האודיו-קליפ הרלוונטי למאורע, בשביל זה ניצור מתודה מיוחדת שתחפש את הצליל מתוך המערך ובמידה וקיים אובייקט כזה גם תנגן אותו, אחרת תעדכן אותנו (המתכנתים) שמנסים להגיע לאובייקט שלא קיים ותחזיר ערך ריק. ל-c# יש פונקציות מיחדות לאוספים, בניהם מתודות ייחודיות למערכים. נוכל להשתמש במתודה `Array.find(array,function)` שמקבלת מערך ומצביע לפונקציה שלפיו היא תחפש. אפשר ליצור פונקציה במיוחד או להשתמש בפונקצית למבדה שמבחינת סינטקס זהה כמעט לחלוטין לג'אווה. נשתמש במרחב השם **`using System;`** לשם כך. במקרה שלנו נדרוש שהמתודה תקבל כפרמטר את קובץ האודיו שאותו אנחנו רוצים למצוא מתוך המערך ובפונקציית למדה נחפש משתנה סאונד שהשם שלו זהה לשם המחרוזת אותה הביאו לנו:

```
public void Play(string name)
    {
        Sound s = Array.Find(sounds, sound => sound.name == name);
        if (s == null)
        {
            Debug.LogWarning("sound" + name + "not found");
            return;
        }
        s.source.Play();
    }
```

בשביל להשתמש ב`Audio Manager` שיצרנו באובייקטי משחק אחרים נוכל: א. במקרים בהם אנחנו אמורים להפעיל כמה אודיו-קליפים דרך אותו אובייקט נשמור משתנה עצם מסוג `AudioManager` ונפעיל את הפונקציה `play` שלו במידת הצורך.
ב)אפשר גם להשתמש בפונקציה של `unity` שמביאה לנו אובייקט שמוצאת לנו אובייקט מסוים ואז נפעיל את `play`:
`FindObjectOfType<AudioManager>().Play(…);`

במדריך על Audio Manager השתמשנו בסרטון הבא:

https://www.youtube.com/watch?v=6OT43pvUyfY&t=679s

של הערוץ Brackeys. הערוץ הוא ערוץ מעולה למפתחי משחקים ובייחוד לאנשים שיש להם ניסיון בתכנות. מומלץ בחום להעשרה בין בפן התכנותי, ובין בפן העיצובי של המשחקים.
לפני שנעבור לפרק הבא אני רוצה להמליץ על האתר:

https://freesound.org/

האתר מספק מאגר עצום של צלילים ברישיון CC שניתן להוריד בחינם.