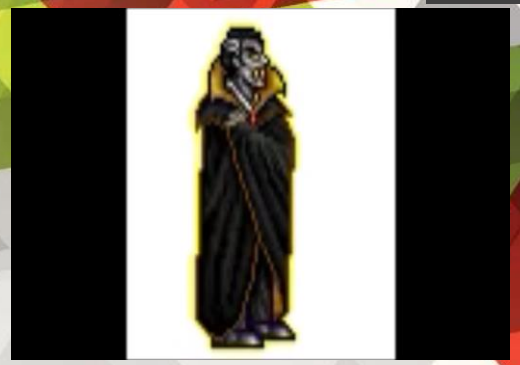
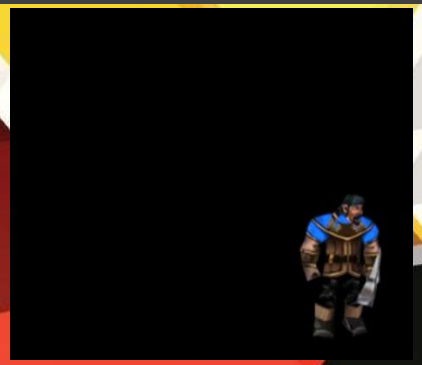



# SOUND IMPORT U UNITY-U I C#

Dejan Seretinek
Mentor : Dominik Cvetkovski



# Teme koje obrađuje prezentacija

**ZVUK U IGRAMA DO DANAS**


**KORIŠTENJE SOUND IMPORTA U UNITY-U**

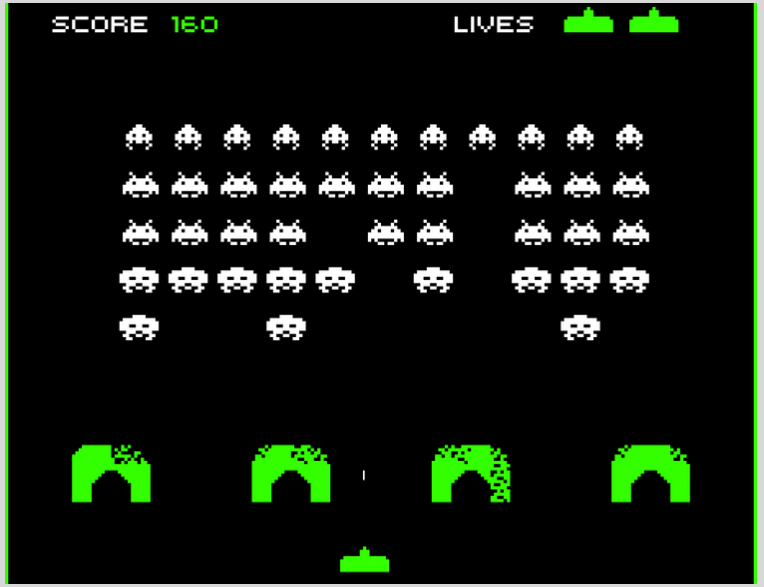

**PISANJE SKRIPTE U C# ZA KORIŠTENJE ZVUKA I ZADATCI**

# Zvuk u igrama do danas

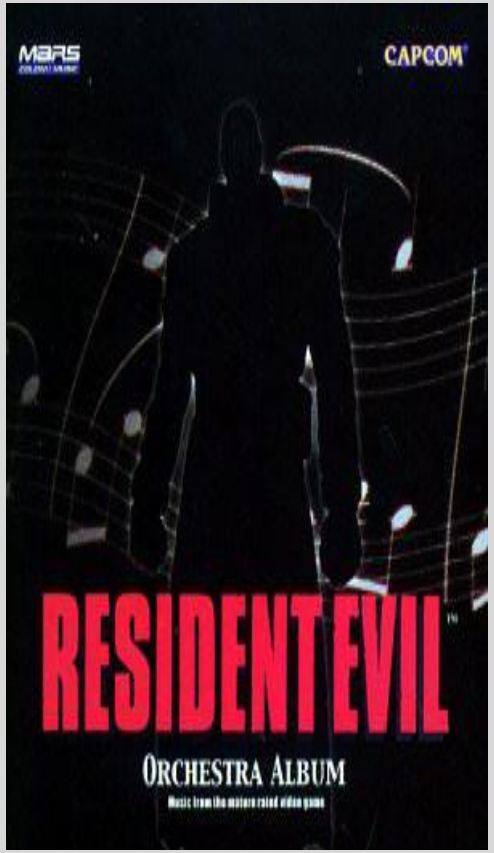
- Dizajn zvuka i glazba igraju važnu ulogu u ukupnom iskustvu modernih videoigara.
- Zvuk igre može pomoći dizajnerima da stvore napetost, dodaju osjećaje, stvore uronjenje u svijet igre, pa čak i riješe probleme s dizajnom.
- Mnoge audio tehnike od kojih danas imamo koristi moguće su samo zbog domišljatosti skladatelja igara i dizajnera zvuka iz prošlosti.
- Pitanje : Što je zvuk i kako nastaje ?
- Programeri su zvuk koristili samo kao efekt, sve do pojave igre „Metroid“ 1980.-ih godina, kada se javlja muzika za video igre (engl. Game music).
- 1983. godine Dave Smith predstavlja MIDI, komunikacijski protokol koji omogućuje digitalnim instrumentima i kompjutorima međusobnu interakciju.

# Pong i Space Invaders

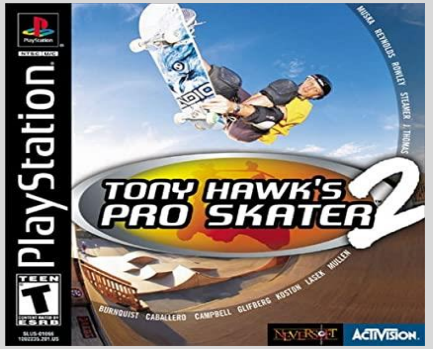
- Sa samo tri jednostavna zvuka, radnje na ekranu međusobno su se razlikovale
- To je doprinjelo opipljivosti događajima u igri.



# Glazba u igrama


# Zvukovi u igrama


- Možemo sve zvukove u igrama podijeliti na četiri kategorije:
  - Ambijent
  - Efekti
  - Dijalozi
  - Muzika

Dalje zvukove možemo podijeliti na linearne i nelinearne.

a. Linearan zvuk se komponira od startne točke i postepeno napreduje do slijedeće točke, a kompozitor ima potpunu kontrolu nad vremenskom crtom određene kompozicije, gdje može stvarati, primjerice, dramatičnu crtu i nastaviti emociju

b. Kod nelinearnog zvuka svaki audio znak mora biti dizajniran sam za sebe (engl. Stand alone), jer je nemoguće predvidjeti sve mogućnosti njegova kretanja. Zvuk za video igre se razlikuje od filma jer je u filmu točno određen slijedeći događaj, dok u igri nije.

1. Interaktivni zvuk (reakcija na Gameplay)
2. Prilagođavajući zvuk  (promjena zvuka koja se adaptira u stvarnom vremenu a nije direktno povezana s korisnikom.
3. Dinamički audio (korisnik ne može birati kada će svirati npr. Promjena zvuka ovisno o dobu dana).

# Glazba u igrama


Još jedna značajna osobina modernog zvuka u igrama je upotreba dinamične glazbe. Igre su postale složene i više ih se ne treba igrati linearno ili imaju potpuno skriptirane događaje, što znači da bi najbolja glazbena igra trebala reagirati na događaje u igri, kakvi god oni bili.

Dizajneri zvuka sada mogu snimati zvuk iz stvarnog svijeta i stvarati biblioteke zvukova koji će se koristiti u njihovoj igri.

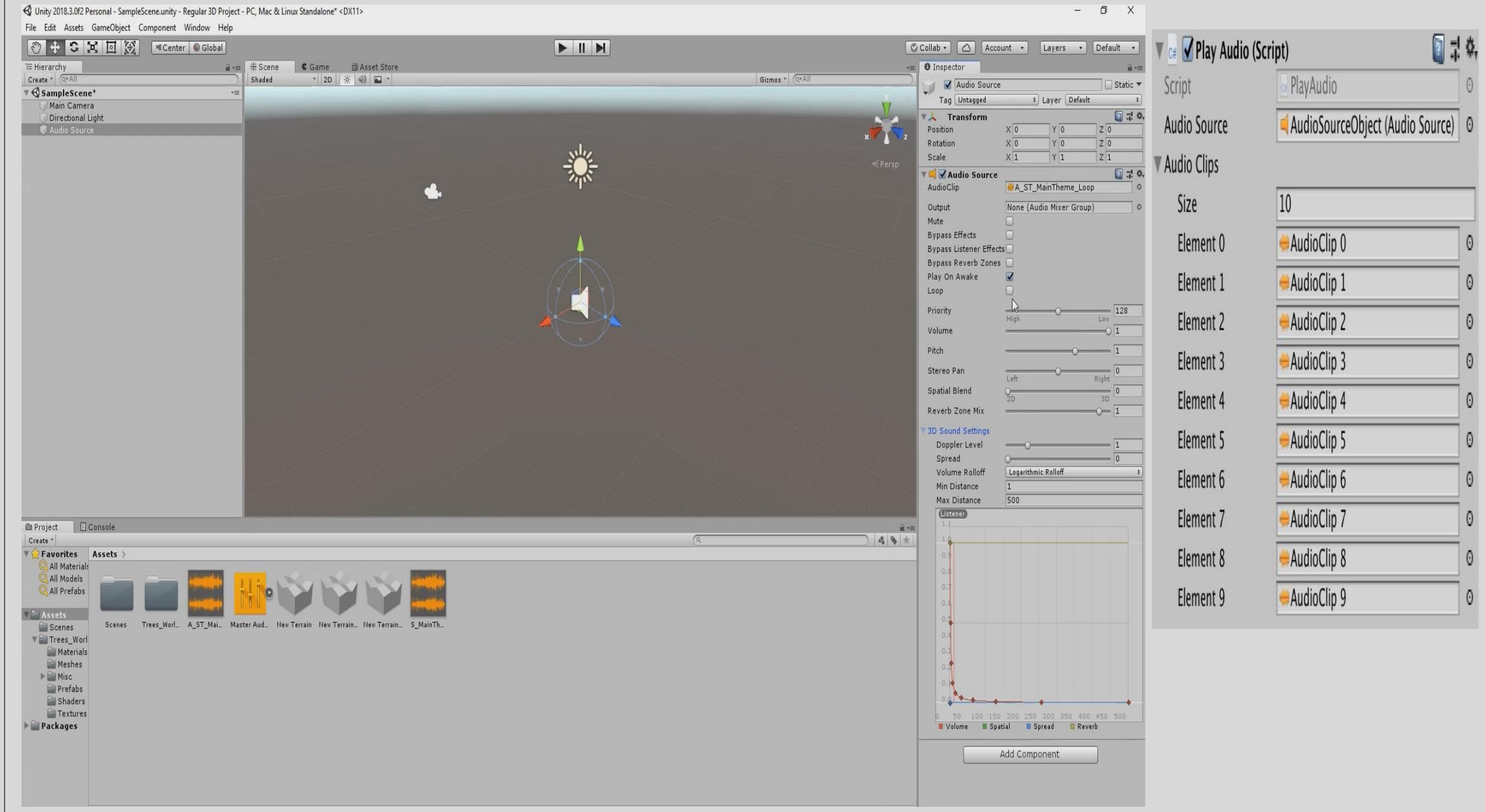
# Prijamnik zvuka i izvor zvuka

**Audio Listener**

- Postavlja se u kameri
- Omogućava igraču da čuje zvukove ovisno o udaljenosti
- Za ispravan rad je potreban jedan prijamnik zvuka.

**Audio Source**

- Drugi dio zvuka je izvor zvuka (eng. Audio Source) koji na određenom objektu pušta zvuk unutar svog dometa u sceni.
- Dodavanje zvuka na Game object on postaje Audio Source
- Zvuk može postići Doplerov efekt ovisno o tome koliko je igrač udaljen od zone izobličenja (eng. Reverb Zones).
- Filteri
- Audio Mixer


Unity 2018.3.0f2 Personal - SampleScene.unity - Regular 3D Project - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
SampleScene*
Main Camera
Directional Light
Audio Source
Scene
Game
Asset Store
Persp
Inspector
Audio Source
Static
Tag Untagged
Layer Default
Transform
Position
Rotation
Scale
Audio Source
AudioClip A_ST_MainTheme_Loop
Output None (Audio Mixer Group)
Mute
Bypass Effects
Bypass Listener Effects
Bypass Reverb Zones
Play On Awake
Loop
Priority 128
Volume 1
Pitch 1
Stereo Pan 0
Spatial Blend 0
Reverb Zone Mix 1
3D Sound Settings
Doppler Level 1
Spread 0
Volume Rolloff Logarithmic Rolloff
Min Distance 1
Max Distance 500
Listener
Volume Spatial Spread Reverb
Add Component
Project
Console
Favorites
Assets
Scenes
Trees_Worl
Materials
Meshes
Misc
Prefabs
Shaders
Textures
Packages
Play Audio (Script)
Script PlayAudio
Audio Source AudioSourceObject (Audio Source)
Audio Clips
Size 10
Element 0 AudioClip 0
Element 1 AudioClip 1
Element 2 AudioClip 2
Element 3 AudioClip 3
Element 4 AudioClip 4
Element 5 AudioClip 5
Element 6 AudioClip 6
Element 7 AudioClip 7
Element 8 AudioClip 8
Element 9 AudioClip 9

# Sound Manager

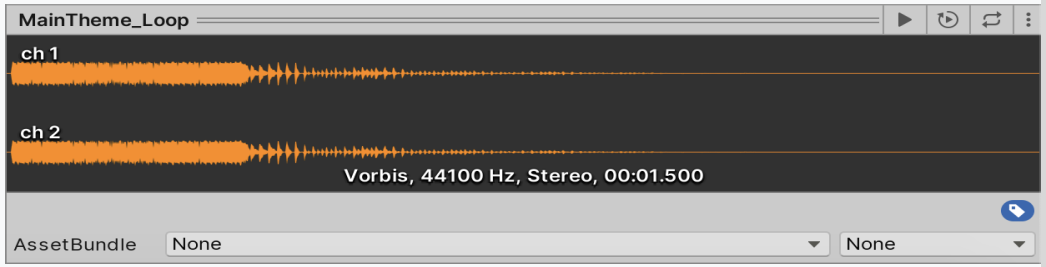
- Podešavanje svih zvukova u igri
- Edit > Project Settings > Audio
- Najjednostavniji način za rad sa zvučnim zapisima u Unity-u
- Kontrolira ostale zvučne zapise.


| Property: | Function: |
|---|---|
| Volume | The volume of all sounds playing. |
| Rolloff Scale | Sets the global attenuation rolloff factor for Logarithmic rolloff based sources (see Audio Source). The higher the value the faster the volume will attenuate, conversely the lower the value, the slower it attenuate (value of 1 will simulate the "real world"). |
| Doppler Factor | How audible the Doppler effect is. When it is zero it is turned off. 1 means it should be quite audible for fast moving objects. |
| Default Speaker Mode | Defines which speaker mode should be the default for your project. Default is 2 for stereo speaker setups (see AudioSpeakerMode in the scripting API reference for a list of modes). |
| Sample Rate | Output sample rate. If set to 0, the sample rate of the system will be used. Also note that this only serves as a reference as only certain platforms allow changing this, such as **iOS** or Android. |
| DSP Buffer Size | The size of the DSP buffer can be set to optimise for latency or performance |
| Default | Default buffer size |
| Best Latency | Trades off performance in favour of latency |
| Good Latency | Balance between latency and performance |
| Best Performance | Trades off latency in favour of performance |
| Virtual Voice Count | Number of virtual voices that the audio system manages. This value should always be larger than the number of voices played by the game. If not, warnings will be shown in the console. |
| Real Voice Count | Number of real voices that can be played at the same time. Every frame the loudest voices will be picked. |
| Disable Audio | Deactivates the audio system in standalone builds. Note that this also affects the audio of MovieTextures. In the editor the audio system is still on and will support previewing **audio clips**, but AudioSource. Play calls and playOnAwake will not be handled in order to simulate behavior of the standalone build. |

# Audio zapisi u Unity –u

- Unity podržava većinu standardnih audio zapisa i omogućava mix i master u stvarnom vremenu . Ima mogućnost dodavanja raznih efekata poput reverba, jeke, kompresije itd.

- Mono, stereo i korištenje do osam kanala
- Mogućnost snimanja zvuka iz gameplaya




Waveform koji obogućava da se presluša audio zapis

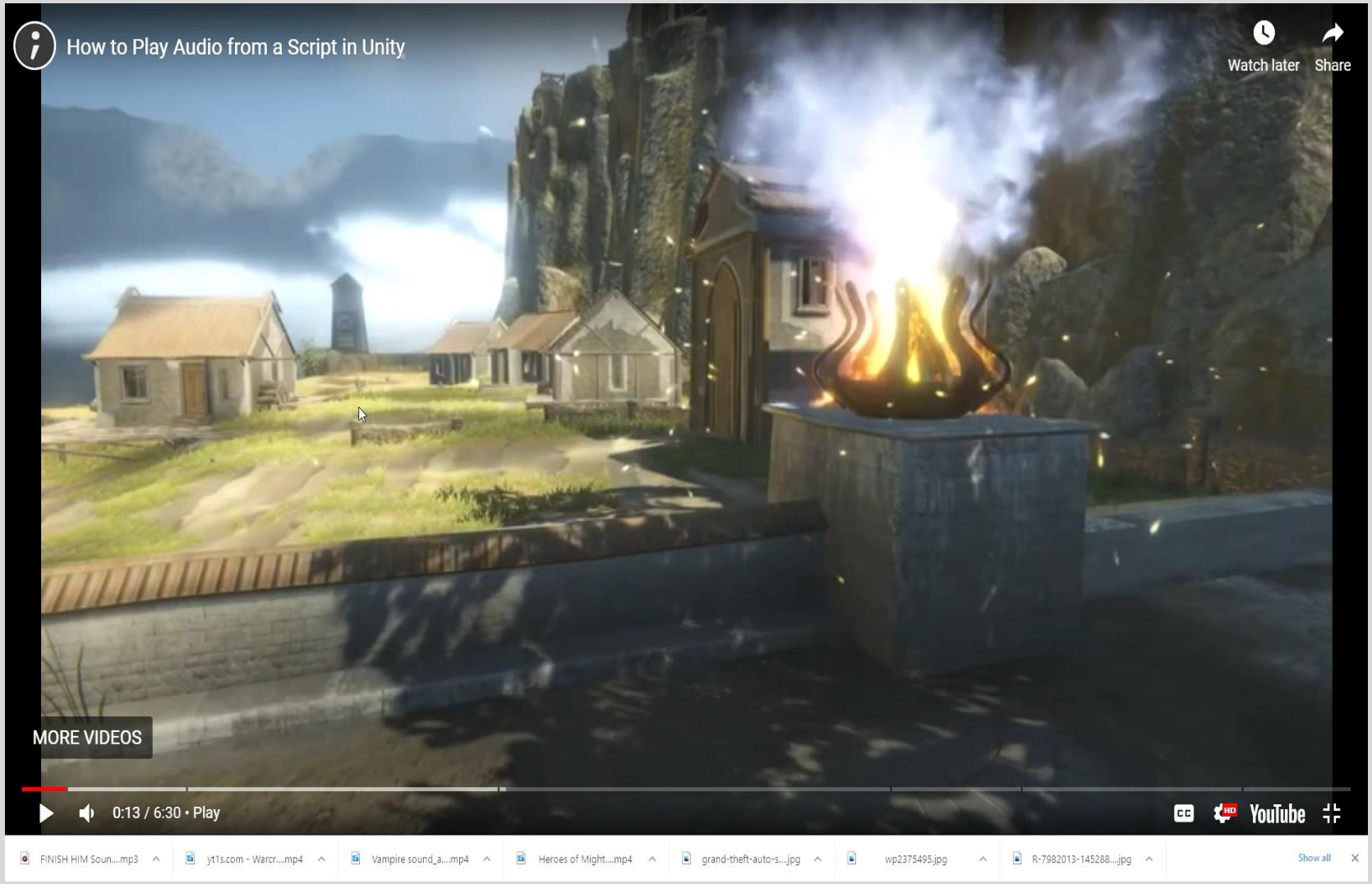
| Format | Extensions |
|---|---|
| MPEG layer 3 | .mp3 |
| Ogg Vorbis | .ogg |
| Microsoft Wave | .wav |
| Audio Interchange File Format | .aiff / .aif |
| Ultimate Soundtracker module | .mod |
| Impulse Tracker module | .it |
| Scream Tracker module | .s3m |
| FastTracker 2 module | .xm |

# Izrada zvuka

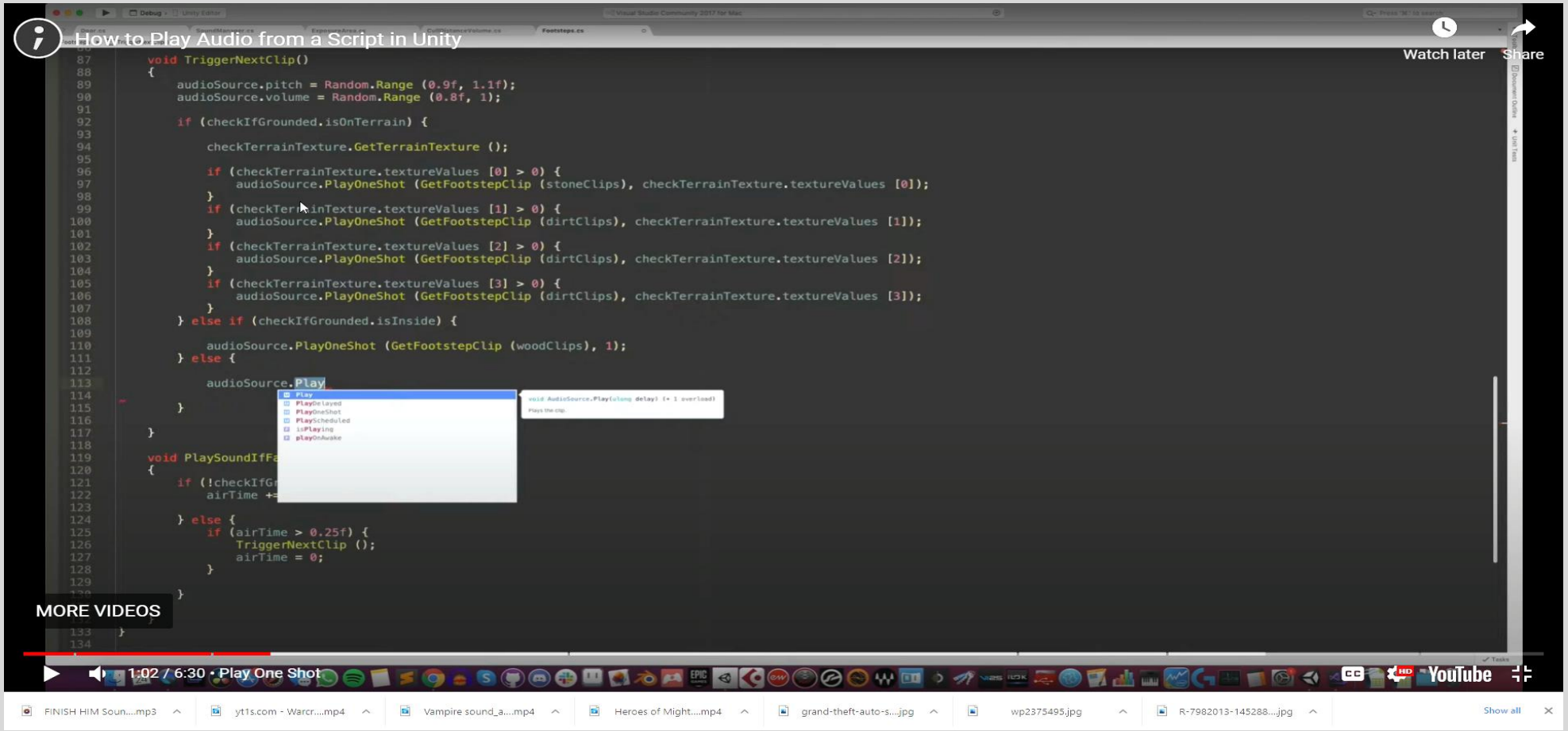
- DAW (Digital Audio Workstation)
- Panorama je pozicioniranje zvuka u zvučno polje. Kako bi se razumjela panorama, treba razumjeti da „stereo“ zvučni sustav predstavlja zvuk prostorno
- Faze izrade zvuka
  - Predprodukcija
  - Produkcija
  - Post - produkcija
- Post produkcija podrazumijeva integraciju zvukova u „Middleware-u“, te izgradnju „builda“, prema izabranoj platformi. Za razliku od ranijih metodologija rada, kada je programer ili audio programer implementirao zvuk u igru, sa čestim posljedicama u vidu pogrešaka i loše dinamike i tranzicije, sada dizajner direktno integrira zvuk u igru i izbacuje „Build“ u programskom kodu

# Najćešće naredbe

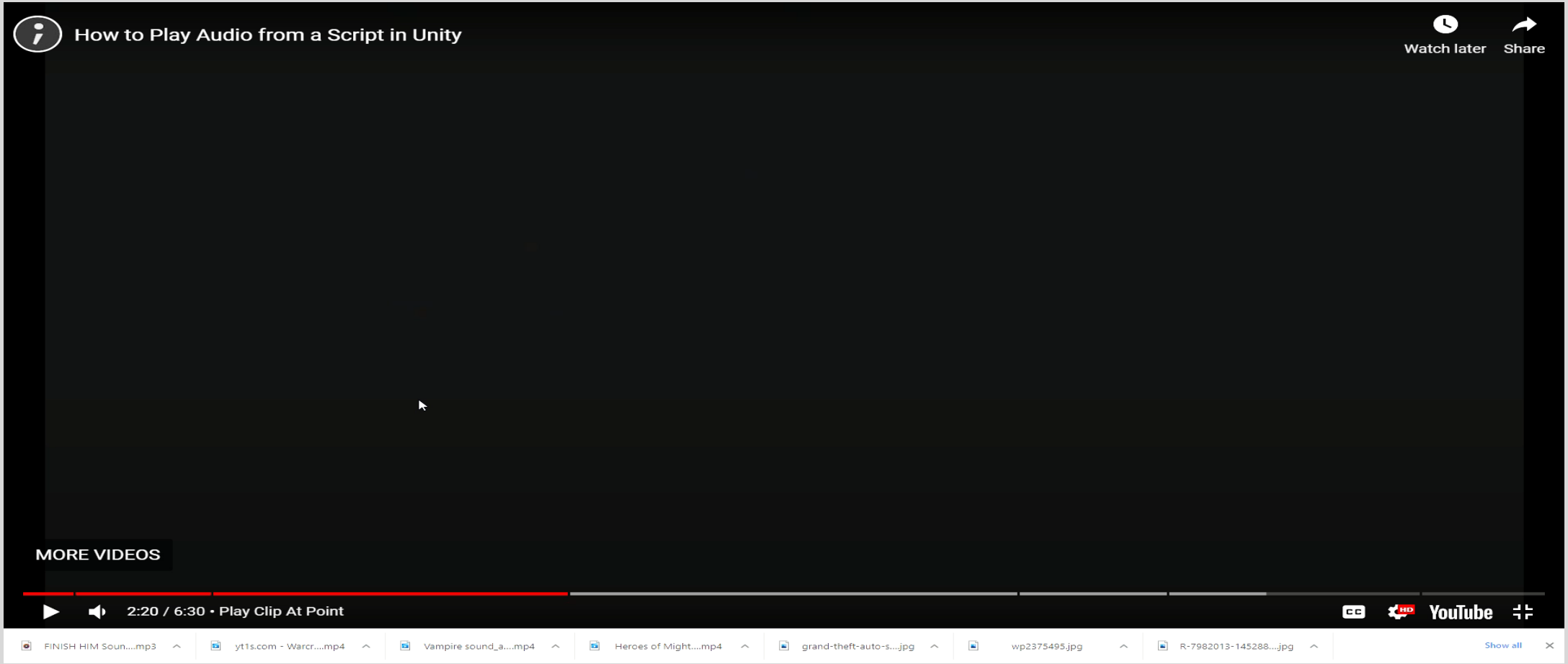
- PLAY
- PLAY ONE SHOT
- PLAY CLIP AT POINT
- PLAY DELAYED
- PLAY SCHEDULED



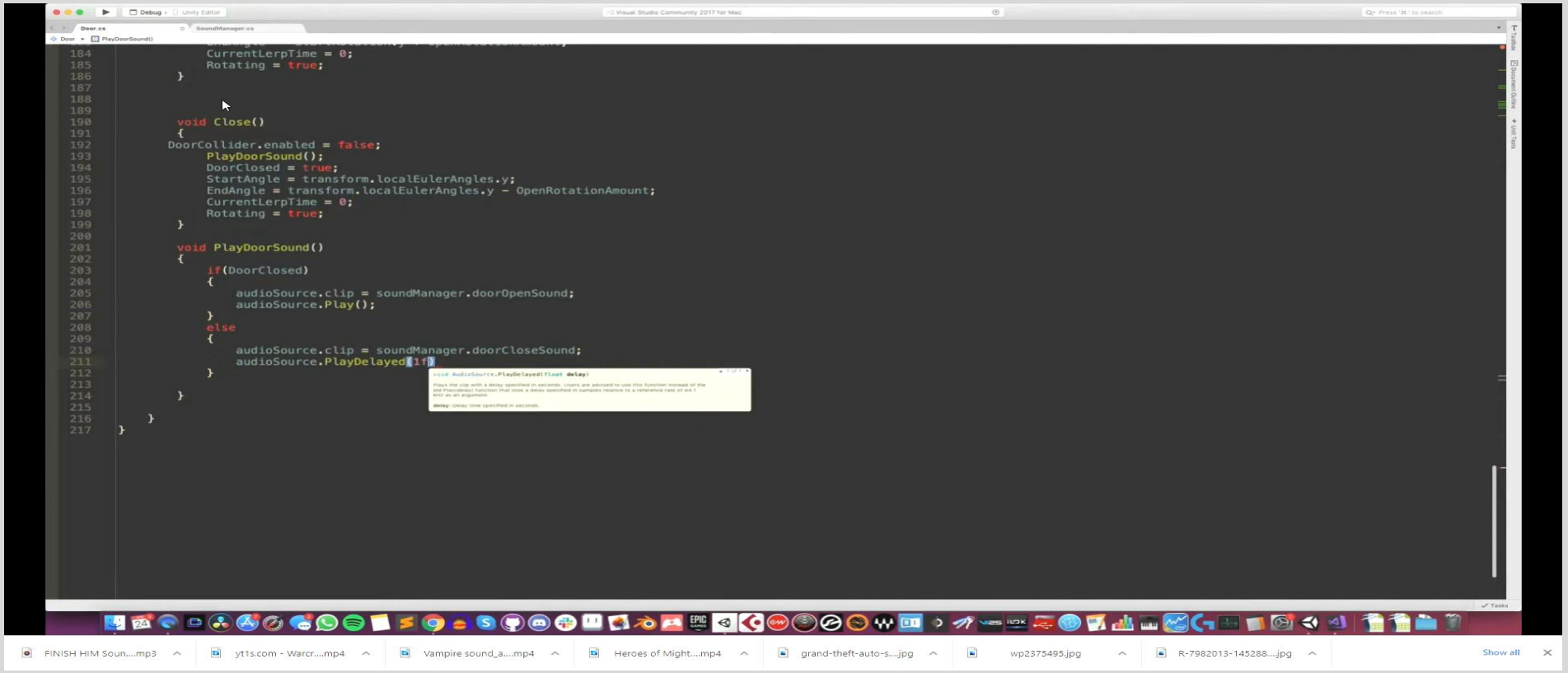
# Play One Shot



# Play On Point



# Play Delayed i Play Scheduled



# Koliko bi Audio Sourceva imali u ovim scenama i za što ?


# Objasnite pojedine skripte

```
public class PlayAudio : MonoBehaviour
{
    public AudioSource audioSource;
    public AudioClip[] audioClipArray;

    public float timeBetweenShots = 0.25f;
    float timer;

    void Update()
    {
        timer += Time.deltaTime;
        if (timer > timeBetweenShots)
        {
            audioSource.PlayOneShot(RandomClip());
            timer = 0;
        }
    }

    AudioClip RandomClip()
    {
        return audioClipArray[Random.Range(0, audioClipArray.Length-1)];
    }
}
```

```
public class PlayAudio : MonoBehaviour
{
    public AudioSource audioSource;

    void Start()
    {
        audioSource.Play();
    }
}
```

```
public class PlayAudio : MonoBehaviour
{
    public AudioSource audioSource1;
    public AudioSource audioSource2;

    void Start()
    {
        audioSource1.PlayScheduled(AudioSettings.dspTime);
        double clipLength = audioSource1.clip.samples /
audioSource1.clip.frequency;
        audioSource2.PlayScheduled(AudioSettings.dspTime + clipLength);
    }
```

```
public class PlayOnCollision : MonoBehaviour
{
    public AudioSource audioSource;

    void OnCollisionEnter(Collision collision)
    {
        audioSource.Play();
    }
}
```

```
public class PlayOnDestroy : MonoBehaviour
{
    public AudioClip audioClip;

    void DestroyObject()
    {
        AudioSource.PlayClipAtPoint(audioClip, transform.position);
        Destroy(gameObject);
    }
}
```

```
private AudioSource audioSource;
public AudioClip[] shoot;
private AudioClip shootClip;

void Start()
{
    audioSource = gameObject.GetComponent<AudioSource>();
}

void Update()
{
    if (Input.GetKeyDown(KeyCode.Space))
    {
        int index = Random.Range(0, shoot.Length);
        shootClip = shoot[index];
        audioSource.clip = shootClip;
        audioSource.Play();
    }
}
```

# Ponavljanje

```
+
+ public class Zadatak_3 : MonoBehaviour
+ {
+     AudioSource audioS;
+
+     private void Start()
+     {
+         audioS = GetComponent<AudioSource>();
+     }
+
+     private void Update()
+     {
+         if(audioS.mute == true && Input.GetKeyDown(KeyCode.K))
+         {
+             audioS.mute = false;
+         }
+         if (audioS.mute == false && Input.GetKeyDown(KeyCode.M))
+         {
+             audioS.mute = true;
+         }
+     }
+ }
```

```
+ public class Zadatak_5 : MonoBehaviour
+ {
+     AudioSource ass;
+
+     private void Start()
+     {
+         ass = GetComponent<AudioSource>();
+     }
+
+     private void Update()
+     {
+         if(Input.GetKeyDown(KeyCode.S))
+         {
+             ass.Play();
+         }
+         if (Input.GetKeyDown(KeyCode.P))
+         {
+             ass.Pause();
+         }
+         if (Input.GetKeyDown(KeyCode.U))
+         {
+             ass.UnPause();
+         }
+         if (Input.GetKeyDown(KeyCode.M))
+         {
+             if(ass.mute)
+             {
+                 ass.mute = false;
+             }
+             else if(ass.mute == false)
+             {
+                 ass.mute = true;
+             }
+         }
+         if (Input.GetKeyDown(KeyCode.S))
+         {
+             ass.Play();
+         }
+         if (Input.GetKeyDown(KeyCode.T))
+         {
+             ass.Stop();
+         }
+     }
+ }
```

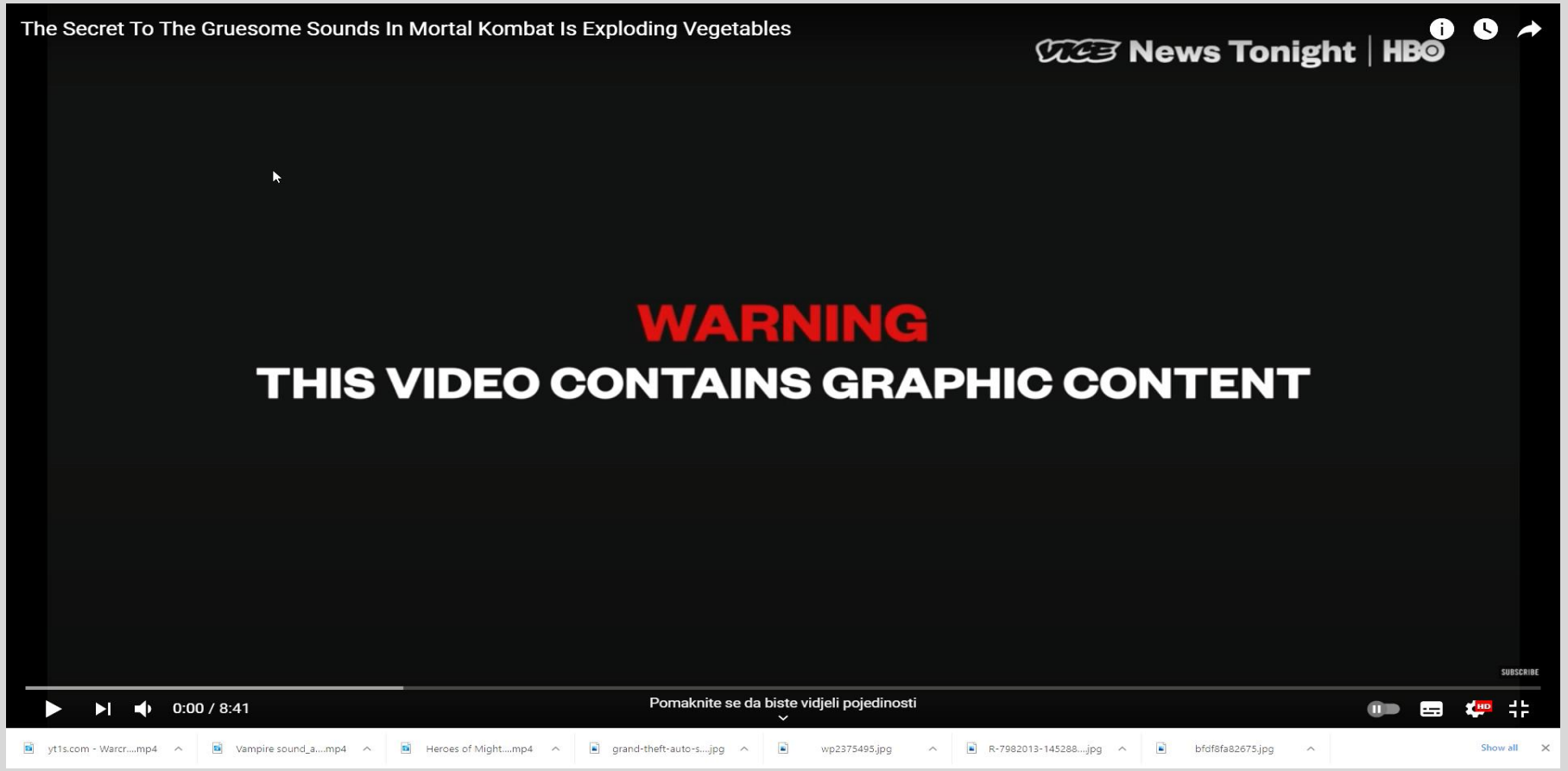
```
@@ -0,0 +1,22 @@
+ using System.Collections;
+ using System.Collections.Generic;
+ using UnityEngine;
+
+ //U updatu palite ili gasite game object ovisno o njegovom stanju konstantno u suprotno
+
+ public class Zadatak2 : MonoBehaviour
+ {
+     public GameObject predmet;
+
+     private void Update()
+     {
+         if(predmet.activeSelf == true)
+         {
+             predmet.SetActive(false);
+         }
+         else if (predmet.activeSelf == false)
+         {
+             predmet.SetActive(true);
+         }
+     }
+ }
```

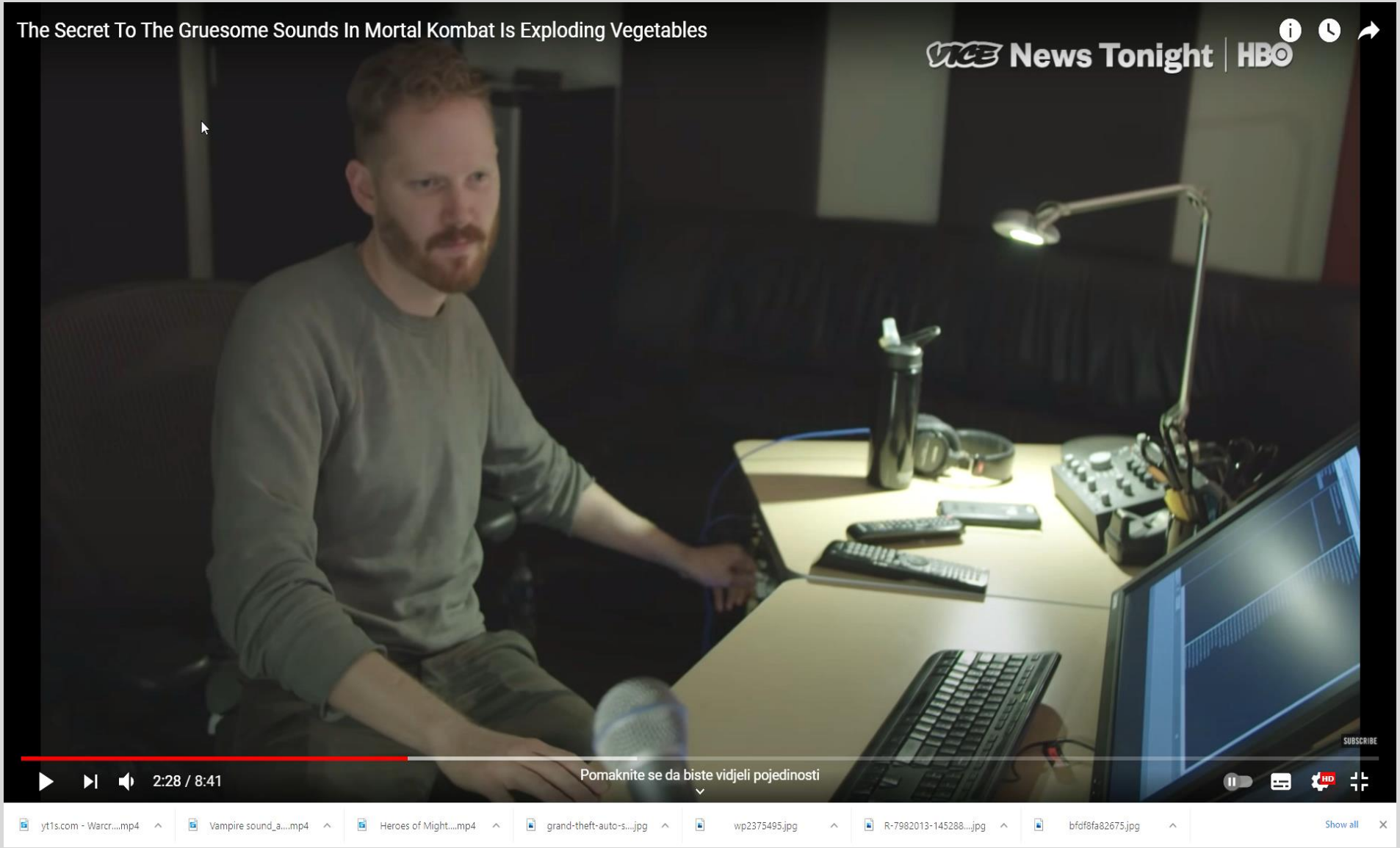
```
+ public class Zadatak5 : MonoBehaviour
+ {
+     public GameObject objekatEin;
+     public GameObject objekatZwei;
+     public GameObject objekatDrei;
+     public GameObject objekatVier;
+     public GameObject objekatFunf;
+
+     private void Update()
+     {
+         if (Input.GetKeyDown(KeyCode.S))
+         {
+             objekatEin.SetActive(!objekatEin.activeSelf);
+             objekatZwei.SetActive(!objekatZwei.activeSelf);
+             objekatDrei.SetActive(!objekatDrei.activeSelf);
+             objekatVier.SetActive(!objekatVier.activeSelf);
+             objekatFunf.SetActive(!objekatFunf.activeSelf);
+         }
+         if (Input.GetKeyDown(KeyCode.K))
+         {
+             objekatDrei.SetActive(!objekatDrei.activeSelf);
+         }
+     }
+ }
```

# Zaključak

- Evolucija razvoja zvuka za video igre.
- Korištenje middlewarea.
- Budućnost razvoja zvuka za video igre
- Najčešće naredbe za sound import u Unity-u
- Audio za video igre
  - Linearni, prilagođavajući i interaktivni

# Hvala na pozornosti :-D




The Secret To The Gruesome Sounds In Mortal Kombat Is Exploding Vegetables
VICE News Tonight | HBO
2:28 / 8:41
Pomaknite se da biste vidjeli pojedinosti
SUBSCRIBE
yt1s.com - Warcr....mp4
Vampire sound_a....mp4
Heroes of Might....mp4
grand-theft-auto-s....jpg
wp2375495.jpg
R-7982013-145288....jpg
bfdf8fa82675.jpg
Show all