Manipulación de audio 3D en python orientado a juegos

Laboratorio de Redes y Sistemas Operativos
Miguel Barraza
Victoria Castro Riquelme
2016

Índice

1. Introducción
   1.0 Audio en 3D: Diferencias con mono, stereo y surround
   1.1 Qué es un audiojuego
   1.2 Lenguaje de programación python
   1.3 Instalación de Python 3.5
   1.4 Audio en Python
2. Librerías elegidas: Cuáles y por qué
3. Ejemplo multimedia
4. Cómo encarar la programación de un audiojuego 3D
5. Implementación propia

7. Fuentes y recursos

1. Introducción

En el proceso de desarrollo de un juego por computadora son muchos los factores a tener en cuenta: diseño, planificación, gráfica, sonido, música, etc, Hay muchos extensos libros que se enfocan en uno o varios de estos temas, en el actual trabajo nos concentraremos en manipulación de sonido orientado principalmente a juegos por computadora.

El audio en los videojuegos se ha instaurado como una progresión natural del mismo modo que llegó al cine o a la televisión, y su uso ha pasado de ser un mero ornamento a formar parte fundamental de la experiencia, llegando al punto de existir títulos centrados exclusivamente en la "diversión acústica".

Si nos centramos en la historia de los videojuegos podemos decir que "suenan" desde que hace más de 40 años se le ocurriera a alguien que las palas del juego Pong debían hacer ruido. A día de hoy, enormes orquestas de renombre realizan conciertos con piezas nacidas en el seno de la industria, y muchísimas canciones forman parte de la cultura popular. La música chiptune, aunque desfasada tecnológicamente, se ha convertido en un ancla del pasado para aquellos nostálgicos que le encontraban sentido y sentimiento a esas viejas formas de onda. George Lucas es un gurú en cuanto a su visión del entretenimiento digital, él decía "El sonido es el 50% de una experiencia cinematográfica". Aunque los videojuegos poseen el factor interactivo como potenciador y distintivo con respecto a una experiencia meramente contemplativa, es bastante fácil acomodar esta frase en su espectro dada la evolución que ha sufrido con los años. Una experiencia interesante es agarrar un título moderno en cualquier plataforma de juego y mutearle el sonido, y es probable que esté faltando un elemento importante para la inversión en el juego.

A lo largo de este trabajo nos concentraremos en desarrollar script para manipulación de audio, mostrando ejemplos que pueden ser utilizados para un juego. Terminaremos mostrando una pequeña implementación de un audiojuego en 3d.

1.0 Audio en 3D: Diferencias con mono, stereo y surround

Tradicionalmente han existido dos métodos para grabar sonido: Mono y Estereo. Para grabar mono se usa un sólo micrófono para captar el sonido, mientras que el esterio utiliza dos (espaciados entre ellos). Cuando escuchamos un sonido mono o monofónico, escucharemos lo mismo por ambos auriculares o parlantes. Al escuchar un sonido estereo, prestando atención podremos notar pequeñas (o no tanto) diferencias entre el auricular/parlante izquierdo y derecho.

Sonido binaural:

El sonido binaural es aquel que, siendo grabado mediante el uso de dos micrófonos, intenta crear para el oyente una sensación de sonido 3D estéreo similar a la de estar físicamente en la habitación o el lugar donde se producen los sonidos.

Audio 3d:

La audición es un proceso complejo. El cerebro humano, para interpretar un sonido, ha de conjugar la información que le llega de ambos oídos.

La información que el cerebro recibe de cada uno de los oídos es diferente —salvo cuando están equidistantes de la fuente—, porque ambos oídos están físicamente separados entre sí por la cabeza. Esta diferencia en la situación de los oídos es la que le permite al cerebro localizar la fuente sonora.

En el sistema auditivo la sensación tridimensional está relacionada con la diferencia de amplitud y tiempo que recibe cada oído. Es decir, la localización de los sonidos en el espacio se consigue con el procesamiento por separado de la información de cada oreja y con la posterior comparación de fase y nivel entre ambas señales.

1.1 Qué es un audiojuego

Un audiojuego es un juego electrónico que se juega en un dispositivo (como una computadora personal). Es similar a un videojuego excepto que el único sistema de realimentación es auditivo en vez de visual.

Los audiojuegos originalmente comenzaron como juegos accesibles PARA LAS PERSONAS NO VIDENTES y fueron desarrollados principalmente por programadores amateurs. Pero el interés por los audiojuegos va en aumento, por parte de artistas del sonido, investigadores de accesibilidad a los juegos, desarrolladores de juegos móviles y jugadores de video. La mayoría de los audiojuegos corren sobre computadoras personales, aunque existen unos pocos audiojuegos para dispositivos portátiles y consolas. Los audiojuegos comprenden la misma variedad de géneros que los videojuegos, tales como juegos de aventura, juegos de carreras, etc.

1.2 Lenguaje de programación python:

En el mercado hay cientos de lenguajes de programación privativos y libres para poder desarrollar juegos offline y online. En la elección decidimos desarrollar con el lenguaje python ya que:
* Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.
* Es multiplataforma : El mismo código funciona en cualquier arquitectura, la única condición es que disponga del intérprete del lenguaje. No es necesario compilar el código una vez para cada arquitectura.
* Tiene una comunidad muy activa: en muchos países se hacen encuentros (conferencias pycon, o encuentros diarios pyday) para compartir experiencias con el lenguaje. En argentina existe la comunidad pyar (python argentina) la cual siempre están dispuesto a ayudar, pueden visitar su web en: www.python.org.ar.
* Paquetes: el lenguaje cuenta con cientos de módulos desarrollado por la comunidad, que aportan código que solucionan muchas de las tareas, En el actual trabajo utilizaremos el **gestor pip**: Es un sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python.

Inconvenientes:
* Lentitud : Los programas interpretados son más lentos que los compilados. Sin embargo con el poder de procesamiento que tienen las computadoras actualmente la diferencia es casi inapreciable.

1.3 Instalación de Python 3.5

Para instalar la última versión de python, en la actualidad es la 3.5.2, se debe ir a la web oficial:
https://www.python.org/downloads/
y bajar la correspondiente a el sistema operativo que usas. Si se trabaja con windows se baja el ejecutable y se instala como cualquier otro programa. Si estamos trabajando en linux sobre ubuntu, debian o derivados se puede seguir los siguientes pasos:
* bajamos desde la web oficial con el siguiente comando desde una terminal:
wget https://www.python.org/ftp/python/3.5.2/Python-3.5.2.tgz
nota: debemos tener instalado el programa "wget". O bien desde el sitio oficial buscamos para nuestra arquitectura linux y lo bajamos desde un navegador.
* Luego lo descomprimimos, si lo bajamos de la terminal utilizamos el siguiente comando:

sudo tar xzf Python-3.5.2.tgz
y luego ingresamos en la carpeta descomprimida:
cd Python-3.5.2
* y luego procedemos a instalarlo:
lo configuramos para la instalación:
sudo ./configure
y lo instalamos:
sudo make altinstall

Make altinstall se utiliza para evitar la sustitución del archivo binario python por defecto / usr / bin / python

Se puede comprobar la versión instalada de python usando el siguiente comando:
$python3.5 -V

Ya estamos listo para usar. Por default poniendo solo "python" en la terminal se usará la que viene preinstalada que suele ser la 2.7, pero si ponemos "python3.5" usaremos la que recientemente instalamos.

Como instalar pip:

Para instalar el gestor de paquetes pip lo podemos ahcer desde su web oficial:
https://pip.pypa.io/
o bien buscar el archivo para instalar, get-pip, que se puede obtener desde:
https://pip.pypa.io/en/stable/installing/#installing-with-get-pip-py
o bajar con el comando:
wget https://bootstrap.pypa.io/get-pip.py
Para instalarlo se hace invocándolo desde python:
python get-pip.py

1.4 Audio en Python

Cuando nos topamos con la necesidad de poder manipular audio con el lenguaje python, con una simple búsqueda en su wiki oficial podemos encontrar muchas:
https://wiki.python.org/moin/Audio/

ALguna de ellas son:

* Nsound: http://nsound.sourceforge.net/
  biblioteca C++ con módulo Python para la síntesis de audio
* pyAudio: http://people.csail.mit.edu/hubert/pyaudio/
paquete para la el manejo de entrada y salida de audio.
* simpleaudio: http://simpleaudio.readthedocs.org/en/latest/
 Reproducción de audio sencilla y sin dependencias para Python 3

Incluso desde pip podemos buscar librerías para audio o sonido con los siguientes comandos:

$pip search audio
$pip search sound

Entre ellas podemos encontrar:

libaudioverse (0.9a8)
- A library for 3D, environmental audio, and synthesis.
- Una biblioteca para 3D, audio ambiental y síntesis.

2. Librería elegidas: Cuáles y por qué

Para el actual trabajo probamos las librerías mencionadas , pero el primer problema topado que varias tienen compatibilidad únicamente para python 2.7, y no funcionan en 3.5. Las que sí funcionan, aceptan reproducción pero no posicionamiento 2d y 3d, la que hemos encontrado que cumple con nuestro objetivo es libaudioverse: tiene funciones multimedia, funciona en python 3.5, y tiene efectos 2d y 3d.

Para instalarlo lo hacemos desde pip con el siguiente comando:

$pip install libaudioverse

3. Ejemplo multimedia:

Utilizando la librería libaudioverse es muy sencillo manipular el audio, instanciando una referencia al servidor de audio y generando un buffer se puede manipular el pich, paneo, y cualquier otra acción con el audio, como la reproducción.

En el primer ejemplo haremos un player de audio, para observar como reproducir y manipular efectos del audio, el reproductor será controlado por linea de comandos, luego de introducir la ruta al archivo de audio a reproducir (formato ogg o wav) podremos usar los siguientes comandos:

* **pause**: para pausar la reproducción.
* **play**: para retomar la reproducción.
* **seek**: para ir a una posición del audio (indicada en segundos), ejemplo para ir a los 15 segundos pondremos: seek 15
* **rate**: para alterar el pitch del audio, inicialmente está en 1, se puede alterar con números float (decimales), ejemplo: rate 0.9 para disminuir el pitch (escucharlo mas lento y distorsionado), o rate 1.1 para aumentar el pitch (acelerarlo y escucharlo mas agudo).
* **quit**: para salir del reproductor.

el código del player es el siguiente:

mediaplayer.py:

```
# inicio del script:

import libaudioverse
import os.path
libaudioverse.initialize()

server = libaudioverse.Server()
server.set_output_device()

print("""linea de comando Media-player.
Introduzca la ruta de acceso a un archivo en un formato compatible con Libsndfile:
normalmente wave o ogg.""")
filepath = input()
filepath = os.path.abspath(filepath)
filenode = libaudioverse.BufferNode(server)
buffer = libaudioverse.Buffer(server)
buffer.load_from_file(filepath)
filenode.buffer = buffer

# Devolución de llamada cuando el archivo finaliza:
```

```
def finished(obj):
    print("reproducción finalizada.")

filenode.set_end_callback(finished)

filenode.connect(0, filenode.server)

commands = """comandos:
play
pause
rate <numero>
seek <segundo>
quit
"""

print(commands)

while True:
    try:
        command = input().split(" ")
        if command[0] == 'quit':
            break
        elif command[0] == 'play':
            filenode.state = libaudioverse.NodeStates.playing
        elif command[0] == 'pause':
            filenode.state = libaudioverse.NodeStates.paused
        elif command[0] == 'seek':
            to = float(command[1])
            filenode.position = to
        elif command[0] == 'rate':
            to = float(command[1])
            filenode.rate = to
    except Exception as e:
        print("Libaudioverse error.  comando no reconocido.")
        print(commands)

libaudioverse.shutdown()

# Fin del script
```

Como se puede observar en el código, con pocas lineas ya podemos contar con un reproductor con control en el audio. Con controlar el parámetro del buffer: filenode.position podemos definir la posición del audio en la cual queremos que continúe, o con filenode.rate le indicamos el pitch que queremos aplicarle al audio.

Segundo ejemplo: Mini juego para probar ejes cartesianos

```
#!/usr/bin/env python
# demostración de un desarrollo de audio en 3d.


import libaudioverse

class Player:
  position=[50,20,0]
  step=1

  def move(self, ruta):
    if ruta == "n":
      self.position[0]+=self.step
      print("al norte")
    elif ruta == "s":
      self.position[0]-=self.step
      print("al sur")
    elif ruta == "e":
      self.position[1]+=self.step
      print("al este")
    elif ruta == "o":
      self.position[1]-=self.step
      print("al oeste")
    else:
      print("destino invalido")

pj=Player()
pj.step=5
print("para moverse utilice las iniciales: n - norte, s - sur, e - este, o  - oeste")
print("busca donde esta la fuente sonora guiandote solo por el sonido. Cuando la logres uvicar centrada en tus auriculares, ganarás el desafío.")
```

```
print("escribe salir para cerrar el juego.")

# inicializamos el sonido:
libaudioverse.initialize()

server = libaudioverse.Server()
server.set_output_device()

world = libaudioverse.EnvironmentNode(server, "default")
world.panning_strategy = libaudioverse.PanningStrategies.hrtf
source = libaudioverse.SourceNode(server, world)

# ruta al sonido:
filepath = "c:/python/r.ogg"
n = libaudioverse.BufferNode(server)
b = libaudioverse.Buffer(server)
b.load_from_file(filepath)
n.buffer = b
n.connect(0, source, 0)

n.looping = True

world.connect(0, world.server)
source.position.value = pj.position
```

7. Fuentes y recursos

Audiojuego - artículo de wikipedia
https://es.wikipedia.org/wiki/Audiojuego
Escucha binaural - artículo de wikipedia
https://es.wikipedia.org/wiki/Escucha_binural
web oficial de python:
www.python.org
comunidad python argentina:
www.python.org.ar
Página oficial de pip:
https://pip.pypa.io/
Librerias de Audio en Python https://wiki.python.org/moin/Audio/
Librerias de Python para juegos https://wiki.python.org/moin/PythonGameLibraries
DearVR professional audio engine https://www.youtube.com/watch?v=34Y0dwVBq4c
EarGames http://www.eargames.com/
Developing a 3D audiogame
http://www.gamasutra.com/blogs/BrianSchmidt/20130617/194489/Making_Ear_Monsters_Developing_a_3D_Audio_Game.php
The Verge:
http://www.theverge.com/2015/2/12/8021733/3d-audio-3dio-binaural-immersive-vr-sound-times-square-new-york (1)
Audio 3D en juegos (juegos en general, con conceptos de geometría)
http://www.michelepirovano.com/pdf/3DAudioInGames.pdf
Game Developers Conference / 3D Audio: Back to the future (charla)
http://www.gdcvault.com/play/1020451/3D-Audio-Back-to-the