LOUISIANA STATE UNIVERSITY

Physics-Based Sound Synthesis for Video Game Environments

Erin N. Bryson[1], Andrew Pfalz[2,3], Marc Aubanel[2], Edgar Berdahl[2,3]

[1]College of Engineering, McNeese State University, Lake Charles, LA
[2]Center for Computation and Technology, Louisiana State University, LA
[3]School of Music, Louisiana State University, Baton Rouge, LA

Introduction

The current mainstream form of sound generation within a Unity game is to use pre-recorded sounds for in-game events, such as a character walking. By utilizing generated C++ code from Synth-A-Modeler (SaM) and Unity's use of Microsoft's .NET framework, it is possible to create an interface within unity that allows users to alter the "physical properties" of the model. This in turn gives greater freedom for altering sound quality.

Synth-A-Modeler[1]

Synth-A-Modeler (SaM) is an "open-source and modular environment for designing physical models" composed of "virtual strings, masses, resonators, and other virtual mechanical objects." These models—which can be used for synthesizing sound, haptic feedback, and/or visual feedback—can then be compiled into a wide range of targets. To allow users to explore the various sounds resulting from the structural change of the virtual objects and their physical parameters, SaM incorporates:

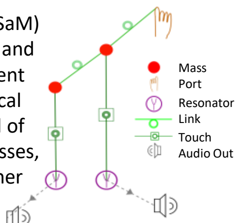


Figure 1: Footstep Model

- Digital waveguides for computationally efficient simulation (Stanford University)
- Model synthesis for allowing resonance frequencies (IRCAM)
- mass-interaction synthesis (ACROE)

Development

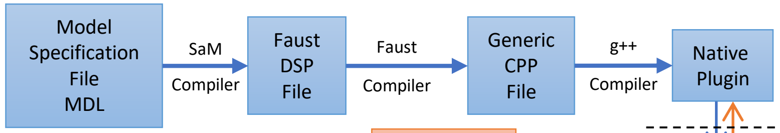
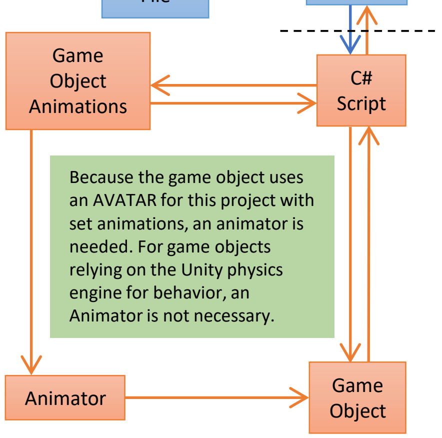


Figure 2: Information Flow From SaM to Unity

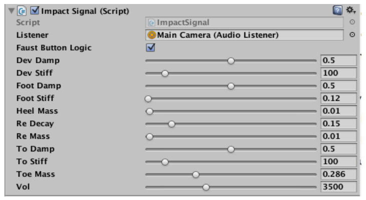
Figure 3: Unity Scene



Because the game object uses an AVATAR for this project with set animations, an animator is needed. For game objects relying on the Unity physics engine for behavior, an Animator is not necessary.

Once the C++ code is placed inside the premade native plugin, a function to alter the physical component values of the "foot" is created and passed to a Unity C# script. This is made possible through Unity's use of the Microsoft .NET Framework.

Both the game object (Figure 3) and its animations draw information from this function via the script interface (Figure 4), allowing users to alter the sound in real time. Once the values are set, the information is sent back to the native plugin, the audio feedback is recalculated, and the new audio data is re-sent into Unity. This communication of information is called marshaling[3].



Impact Signal (Script)

Script	ImpactSignal
Listener	Main Camera (Audio Listener)
Faust Button Logic	✓
Dev Damp	0.5
Dev Stiff	100
Foot Damp	0.5
Foot Stiff	0.12
Heel Mass	0.01
Re Decay	0.15
Re Mass	0.01
To Damp	0.5
To Stiff	100
Toe Mass	0.286
Vol	3500

Figure 4: Custom Interface for Physical Parameter

Discussion

The plugin used for this project is a modified version of the official AudioPluginDemo from Unity; the documentation and download link for the original version can be found in the "Unity Native Audio Plugin SDK" section of the Unity Manual[2]. Our modified plugin is built for the Mac OSX specifically.

Current Limitations:

- Interface must be created manually within a game object's C# scripting component
- Physical models are made outside of Unity, making the utilization of such models require more steps
- Unity does not except DSP files, forcing users to compile them into a CPP file and place the C++ code into a plugin for Unity to draw from

Future Goals:

- Automate user interface creation
- Integrate the graphical interface of Synth-A-Modeler into Unity, which would allow users to utilize these models in fewer steps.
- Set up Unity so that it is able to accept and utilize DSP files

References

1. Synth-A-Modeler Syntax For Version 1.0, Berdahl, Edgar, 4 July 2015.
2. https://docs.unity3d/Manual/
3. https://msdn.Miscrosoft.com/en-US/

Acknowledgements

This material is based upon work supported by the National Science Foundation under award OCI-1263236 with additional support from the Center for Computation & Technology at Louisiana State University.

Center for Computation & Technology