Lab 15 – XNA Audio

Instructions: Complete each problem. If you're struggling with a problem, feel free to ask questions on the class forum.

This lab is optional, but it gives you valuable programming experience. You should definitely complete the lab if you can.

Problem 1 – Create a MonoGame project and build and add audio content

Start up the IDE and create a new MonoGame Windows Project named Lab15. Save the project in a reasonable location on the computer.

Download or create 4 different wav files and name them upperLeft, upperRight, lowerLeft, and lowerRight. Use the Pipeline tool to build them as xnb files and add the xnb files as content to your project.

Mac/Linux users: As of this writing, the Pipeline tool fails when trying to build xnb files from wav files. Although you could use the process in Section B.3. to build the audio content on a Windows machine and then copy it over, there's an easier way. Add an audio folder to the Content folder of your project. Copy the wav files into the new audio folder using the standard file manipulation functionality of your operating system. In your project, right click on the audio folder, select Add > Add Files ..., select all the wav files, and click the Open button. Select all the wav files in the audio folder in your project. Right click and select Build Action > Content. Right click again and select Quick Properties > Copy to Output Directory. You've now got the wav files ready to use in your project.

Problem 2 – Load audio content

Add four `SoundEffect` fields at the top of the `Game1` class to hold your sound effects. You'll need to add a using statement for the `Microsoft.Xna.Framework.Audio` namespace to get this to compile.

In the `Game1` `LoadContent` method, add code to load the four sound effect assets into your four new fields.

Problem 3 – Play sound effects for left clicks on screen

Make the mouse visible in the `Game1` constructor.

In the `Game1` `Update` method, add code to get the current mouse state.

In the `Game1` `Update` method, add an if statement to check whether the left mouse button is pressed. If it is, play the appropriate sound effect based on which quadrant (upper left, upper right, lower left, or lower right) the mouse is in.

This may seem to work if you click the mouse normally. Now just hold the left mouse button down – the game starts playing the sound effect on every update, leading to lots of nasty overlap. Let's fix this to actually use mouse clicks, not mouse button presses.

Add a field to the `Game1` class to tell whether a mouse click was started and initialize the field to `false`.

In the `Game1` `Update` method, add an if statement right after you get the mouse state. If the left mouse button is pressed and the click started flag is `false`, set the click started flag to `true`. Otherwise (else if), if the left mouse button is released and the click started flag is `true`, set the click started flag to `false` and move the code that plays a sound effect based on the quadrant into this else if clause.